Appendix B

PSM-500/500H/500L Remote Control Command Protocol

This Appendix applies to Datum Systems' PSM-500, M500 Class Modem series including the PSM-500 (70 MHz IF), the PSM-500L (L-Band IF) and the PSM-500H (Hybrid 70 MHz/L-Band). The M500 Class Modem is distinguished from M5 Class Modems which include the PSM-4900, PSM-4900H and PSM-4900L. The remainder of this document refers only to these modems as the M500, PSM-500 or the modem.

Revision History

Rev. 0.82, September 19, 2006 – Preliminary -corrections and M5/SnIP compatibility.
Rev. 0.83, October 11, 2006 – Unit Status missing Feature Set string + minor -corrections.
Rev. 0.84, October 16, 2006 – Added BUC and LNB commands.
Rev. 0.85, October 20, 2006 – Added IF/RF Frequency ranges.
Rev. 0.86, November 6, 2006 – SW Rev 0.20 - Added redundancy bits and corrected loop flags.
Rev. 0.87, November 29, 2006 – SW Rev 0.21 - Corrected SnIP command C4 flags.
Rev. 0.88, December 5, 2006 – SW Rev 0.24 – Added Viterbi, Rate ¾, 16QAM, CT to Table A.
Rev. 0.89, December 7, 2006 – SW Rev 0.26 – Added R-S CT220,200 mode.
Rev. 0.90, December 16, 2006 – SW Rev 0.28 – Added section on MCC far end control.
Rev. 0.91, January 4, 2009 – SW Rev 0.77 – Added LDPC and new SnIP control information.
Rev. 0.92, May 12, 2010 – SW Rev 1.17 – Corrections to command 1 store and recall.
Rev. 0.93, Dec 12, 2010 – SW Rev 1.19 – Add Unit Polling Command.

Differences From Previous Datum Systems' Command Protocols

The command protocol described here and used in the M500 Class Modems uses the same basic packet structure as previous versions, but the contents, methods and scope of commands differ significantly. In short the M4 Class Modems used a single packet for each possible function command, while the newer M5 and M500 structure uses a single packet for to control multiple related functions. This was done for two reasons; first, the M500 contains significantly more programmability and second, this new structure improves efficiency in typical applications.
Some other differences between theM500, M5 and M4 protocols are.

- There is no ASCII Packet Protocol in the M5 or M500 Modem.
- The M5 and M500 Modems are capable of accepting commands at the near end for monitor/control of the far end modem. This feature requires that M5 modems be equipped with the "Multiplexer" or "R-S/Multiplexer" option card. The M500 includes the Multiplexer as standard.
- There is no read, write and write to EEPROM function in the M5 or M500 Modems. This is because the non-volatile memory is a different type and is always written to when power is removed from the modem.
- The "Mode" Byte has a different value in all three protocols. It is now used to determine if a command is for the near end or far end modem, and the type of modem. M5 class modems use a "1" mode value for near end control and "2" for the far end. M500 class modems use a "5" mode value for near end control and "6" for the far end.
- All "Read" or request type commands are designated by setting the data byte count to zero.
- Beginning with software version 0.16 the M500 command set includes M5 protocol compatibility limited to the M5 series capabilities. M5 compatible commands simply use the M5 Mode Byte and receive M5 compatible responses. Thus a mixed system of M5 and M500 modems can all be controlled with the same protocol, provided that none of the new M500 capabilities not in the M5 modem are needed. See also the section below titled "Far End Modem Control via the MCC".

M500 Modem Control Overview

The PSM-500 modem can be controlled by the front panel or from an external device. External devices may be a "dumb terminal" or terminal emulation program, a specialized controller connected to either the modem's rear panel USB connection or the rear "Control" DB9 Port. The use of an external "terminal" for

control of the PSM-500 is covered in the main manual. This Appendix describes the protocol for control of the modem by an external computer or controller connecting to either the rear panel DB9 Control Port or the rear panel USB port.

The controlling computer may take many forms ranging from an handheld/portable USB capable device or dedicated stand-alone processor to a personal computer or a larger mini or main-frame computer, and is referred to here simply as a controller. The PSM-500 contains full software allowing it to be externally controlled. Note however that no software is provided for the external controller, which is the responsibility of the user.

The protocol described here contains “place holders” within the data structure in two types, “Reserved” and “Future Expansion”. The reserved data structure elements are used by the factory for specialized testing and calibration. The future expansion elements are spares and data needed for yet to be implemented features. The expanded frequency control needed in L-Band IF type modems were once “future” items.

Setup For Remote Control

Before the PSM-500 can be externally controlled it must be set to operate in the proper remote control mode via the front panel. Several parameters must be set as dictated by the control system to be used:

1. Modem Address (all control methods)
2. Control Interface as either RS-232 or RS-485 (Control Port, J6)
3. USB Enable (USB port only)
4. Bit Rate, Format (all control methods, separate entries for USB and Control Port) – default and standard are 9600, N,8,1.
5. Remote Protocol (Control Port) – must be set to “Binary Packet”.

Front Panel setting of these parameters is available in the <Unit: USB> and <Unit: Remote> columns. These parameters may also be set via the remote control port itself, but this is dangerous, as it will probably result in loss of communications.

The RS-232 interface is only useful in a point to point control with one controller and one modem because of the nature of RS-232. The 485 type interface allows multiple modems and controllers to be tied to the same serial bus. The modem address insures that the modem only responds to messages intended for it.

The RS-485 interface on the modem is configured as a “4 wire” interface. That means that the transmit and receive wire pairs are separate. This allows a controller to both talk and listen at the same time. If a “2 wire” configuration is desired, the transmit and receive pairs may be simply tied together external to the modem. Care should be taken here to insure that the “A” or “-“ side of the transmit is tied to the “A” or “-“ of the receive, and the same for the “B” or “+” side.

Packet Protocol Basics

All remote control communications are formatted as “packets” of information. The packets contain a “header” and a data or “payload” section. The header is more like a wrapper around the data and includes flags, addresses, commands, control information and a data byte count at the front of the message and a closing flag and checksum at the end. The data field can be 0 to approximately 180 bytes depending on the particular packet. A zero data byte count is only used to request information.

The modem never initiates transmission of a packet on its own, it only responds to a request or command packet from the controller. The sequence of events in this protocol is for the controller to send a command packet to a particular addressed modem. The addressed modem reads the command packet and if valid executes the command and sends back a response packet. A response is always returned unless:

a) The unit is improperly addressed, which causes the modem to never see the packet, or
b) The message is globally addressed to all modems, or
c) The message flags or checksum are incorrect causing the modem to reject the message.

If the message packet address is accepted by the modem but the packet format is incorrect then an invalid message response is returned. The response may take one of several formats depending on the command type, but the response format for any particular command is fixed.

Example of Binary Packet Control System

An example Binary control system might consist of a single PC type computer communicating with one to 10 or more modems using an RS-485 interface card installed in the PC as one of the "Com" channels. This setup might be used to monitor and control a small station. The PC could in turn be communicating with a central computer system via a telephone line and modem. A program written in "C" or "BASIC" could periodically request status of each modem to insure that nothing has changed, and upon command from the central computer would change the parameters of any individual modem. For an example of the message format similar to this, see the Binary Packet Command and Response Message sections below.

Binary Packet Command Message Format

The Binary Packet from the controller to the PSM-500 Modem adheres to the following message format.

Byte 0 Optional Pad Byte FF hex	Byte 1 Opening Flag A5 hex	Byte 2 Destination Address 8 bits	Byte 3 Source Address 8 bits	Byte 4 Binary Command 8 bits
Byte 5 Mode Byte 8 bits	Byte 6 Data Byte Count	Byte 7 - (n-3) Data Bytes 180 maximum	Byte n-2 Closing Flag 96 hex	Byte n -1 Checksum	

Address Field

The modem is assigned an address via the front panel control or via the remote control line itself. Modems are normally shipped with the address preset to "1". When multiple modems are connected to the same RS-485 control line each must have a unique address to avoid conflicts. A modem may have the same address as any controller device on a shared bus (not recommended), but no two controller devices may have the same address. Modems respond only to incoming messages containing their unique address in the destination address position of the control message. A destination address of 255 (0xFF) is a global address received by all modems.
The Source address may be any value from 0 to 254 that is not assigned to a modem and becomes the destination address of the response message. This allows for multiple controllers in one system. The convention of using 255 as the global address is assumed here also for controllers. The modem makes no use of the source address other than to place it in the response packet directing the response to the originating command source.

We tend to use controller addresses in the range 0xC0 to 0xCF, simply as an easy to remember mnemonic for "Controller" 0 to 15.
When using the USB control port there are no addresses used or recognized in the packets in either direction. This is because the USB protocols insure delivery to and from the proper destination.

Mode Byte Field

The Mode Byte is "05" for a local (near end) command, "06" hex for commands intended for the far end (remote) modem. Access to remote modem requires that each modem be equipped with the Datum Multiplexer and that it be enabled. Note that the mode byte is not returned in the response packet.

Command Byte Field

The Binary Command Byte is taken from the Command Tables below. Note that there may be multiple command byte tables depending on the modem software version number. The software revision is read from the front panel LCD display.

Data Byte Count Field

The Data Byte Count field includes the total number of Data Bytes only, and should be zero (00) for read mode. Note that this determines if the command is a read (request for information) or write (command to change parameters). The data byte count can only be one of two values, either "0" as is common for a read request, or the actual fixed number of data bytes in that particular packet type. If the structure used in a program to store the contents of a packet's parameters is "packed" so that each element occupies the same number of bytes as the protocol definition, then the data byte count is equal to the "sizeof(structure)" for example in C.

Data Byte Field

Multiple data formats are used within the data field: 1 byte entries are a single character or unsigned byte type or containers for bit flags; 2 byte numbers and 4 byte numbers in both signed and un-signed format. Strings consist of multiple consecutive bytes or characters, and are terminated by a "0" value byte. An extended 6 byte long integer is used for L-Band and RF Frequencies.

No floating point numbers are used, although the incremental value of an entry may allow a decimal point value. For instance the transmit power level is entered as an integer in increments of 0.1 dB, so an entry of -176 represents -17.6 dB. The incremental value (represented by 1 least significant bit change) is determined from the Write Bytes section of the Command Tables by ignoring any decimal point and using the number of displayed digits. Thus frequencies are entered in 1 Hz increments, data rates in 1 bps increments, and times in increments as shown in the tables. No offsets are used in any of the number entries.

Checksum Field

The checksum is computed as 256 minus the sum of all bytes excluding opening and closing pad bytes, and the checksum itself. The checksum is modulo 256, that is the checksum never exceeds 255 in value but rolls over at 256. The sum of all bytes (modulo 256) including the checksum itself is always zero.

Pad Bytes

The Pad Bytes are normally not used, and are not checked by the processor. Multiple pad bytes may be used. Pad bytes are all 1's or "FF" hex. The pad bytes serve several functions in an RS-485 configured system, indicating clean transitions from idle to active states. The M500 modem does not send a starting pad byte in RS-485 responses.

Binary Packet Response Message Format

The Binary Response Packet from the PSM-500 Modem to the controller adheres to the following message format. The response from a modem will occur within approximately ½ second. Note that a modem set to 485 control port mode mutes its receive while sending the response message, so if the 485 bus is configured as 4 wire or 2 wire a modem will not receive a message while responding to a previous message.

Byte 1 Opening Flag 5A hex	Byte 2 Destination Address 8 bits	Byte 3 Source Address 8 bits	Byte 4 Binary Command 8 bits	Byte 5 Status Byte 8 bits
Byte 6 Error Byte	Byte 7 Data Byte Count	Byte 8 - (n-3) Data Bytes 128 maximum	Byte n-2 Closing Flag 96 hex	Byte n -1 Checksum

The Destination address is taken from the Source address of the incoming packet to which this is a response. The Status and Error Bytes are defined later in this Appendix. The Data Byte Count field includes the total number of Data Bytes only, and should never be zero for a response message. The Pad Bytes are normally not necessary and are not checked by the processor, and multiple pad bytes may be used. The checksum is 256 minus the sum of all bytes excluding opening and closing pad bytes, and the checksum itself. The sum of all bytes except pad bytes including the checksum itself is always zero.

Data Payload Format

There may be 0 data bytes in a simple controller request for information, but all other packets have a non-zero data byte count and the corresponding number of bytes of information representing the data sent or received. The data field contains a fairly regular format consisting of Records (Pascal) or Structures (C) with multiple elements. One of the main purposes of this Appendix is to define those structures for each possible data field. The particular data field structure to be used in a message packet is defined by the Command number. A response packet therefore consists of the header and the data structure as defined in the header's command byte.

The standard data structure contains first a field of bit flags (usually 32) that are used in a response to indicate values that have changed since the last read. These same change bit flags are used in a command to change information as "write enable" bits to indicate which particular value is to be changed. A command message must contain the full field of data, but the modem only pays any attention to the values with their corresponding write enable flag bit set.

Each control source has its own set of change flags to determine if the changed data has been read or not. i.e. the USB, front panel, remote control port J6 and the Modem Control Channel (MCC).

Next in the data structure are a field of bit flags used to specify the value of parameters that can only take on a few possible values, for example 1 bit is assigned to indicate if the transmit output is enabled. The parameter bit field is followed by the byte and multi-byte parameters that may be numbers or strings.

Example Binary Packet Command and Response Messages

As an example of using the remote control assume that the controller should check the transmit frequency, and if not set correctly change it to 74.652 MHz. The modem address is set to 12 decimal and the controller is address 200 decimal. Setting the transmit frequency in this example will consist of three operation; 1) Checking the current frequency, 2) Setting the transmit frequency and 3) Checking the response to insure that the setting was correctly accomplished. What is not shown in any of these steps are the necessary background work such as building and receiving packets, calculating checksums and testing for errors.

1. **Check the transmit frequency:**

To read the transmit IF frequency we send a command request to the modem with the Mod IF command number (41 hex) and set the data byte count to zero.

Command packet from controller to read the transmit frequency

Byte 1 Opening Flag "A5 hex"	Byte 2 Destination Address- 8 bits "01 hex"	Byte 3 Source Address- 8 bits "CC hex"	Byte 4 Binary Command 8 bits "41 hex"	Byte 5 Mode Byte 8 bits "05"	Byte 6 Data Byte Count "00 hex"
Byte 7 Closing Flag "96 hex"	Byte 8 Checksum "B2 hex"				

Response packet from modem

Byte 1 Opening Flag "5A hex"	Byte 2 Destination Address- 8 bits "CC hex"	Byte 3 Source Address- 8 bits "01 hex"	Byte 4 Binary Command 8 bits "41 hex"	Byte 5 Status Byte 8 bits "76"	Byte 6 Error Byte "00 hex
Byte 7 Data Byte Count "24 hex"	Byte 8 - (n-3) Data Bytes 128 maximum See Below	Byte n-2 Closing Flag "96 hex"	Byte n -1 Checksum "6A hex"		

Here in compressed format is the exchange between the controller with address 0xCC and a modem with address 1. The first hexadecimal character before the ":" is the byte number.

```
Send:  a5 01 cc 41 05 00 96 b2 To Unit 1

Received 45(0x2d) bytes, Data count 36(0x24)
  00: 5a cc 01 41 76 00 24 00 00 00 00 02 00 08 00 80
  10: 1d 2c 04 00 00 00 00 00 00 9c ff 3c 00 32 00 a2
  20: fe 00 00 00 00 00 00 00 00 00 00 96 e8
```

Expanding the data portion of the response which starts at byte 7 above is:
Byte 0-3 – 00 - hex – Change Flags - No Changes since last read
Byte 4 – 02 hex – Carrier disabled, QPSK mode
Byte 5 – 00 hex – No Preamble, No Burst
Byte 6 – 08 hex – AUPC Off, Mute Automatic, 75 Ohm
Byte 7 – 00 hex - Spares
Bytes 8-11 – 80 1D 2C 04 hex – Frequency = 70.000000 MHz (Hex = 04 2C 1D 80).
Bytes 12-13 – 00 00 hex – Used for L-Band
Bytes 14-17 – 00 00 00 00 hex – No Carrier Offset
Bytes 18-19 – 9C FF hex – Minus 10.0 dBm output level (FF 9C Hex = -10.0 in 0.1dB steps)
Bytes 20-21 – 3C 00 hex – AUPC Eb/No setting (00 3C Hex = 6.0 in 0.1dB steps)
Bytes 22-23 – 32 00 hex – AUPC Max Cxr level allowed (00 32 Hex = +5.0 in 0.1dB steps)
Bytes 24-25 – A2 FE hex – AUPC Min Cxr Level. (FE A2 Hex = -35.0 in 0.1dB steps)
Bytes 26-35 – 00 00 00 00 hex – Spare

This response said that the modem was set to 70.000000 MHz, so the IF frequency parameter must be changed. Note that only the packet byte sequence is given below.

2. **Program the modem transmit frequency.**
Here is the exchange between the controller and a the same modem with address 1:

```
Send To Unit 1:
  00: a5 01 cc 41 05 24 01 00 00 00 02 00 08 00 60 19
  10: 73 04 00 00 00 00 00 00 9c ff 3c 00 32 00 a2 fe
  20: 00 00 00 00 00 00 00 00 00 00 96 ea

Received 45(0x2d) bytes, Data count 36(0x24)
  00: 5a cc 01 41 76 00 24 05 00 00 00 02 00 08 00 60
  10: 19 73 04 00 00 00 00 00 00 9c ff 3c 00 32 00 a2
  20: fe 00 00 00 00 00 00 00 00 00 00 96 c0
mif Change Failed!
```

Command packet from controller to set the transmit frequency to 74.652 MHz:

Packet Start - "A5 0C C8 41 01 24"
Data Bytes:
Byte 0 – 01 hex – Change the Transmit Frequency Only
Byte 1-3 – 00 hex – No other changes
Byte 4 – 02 hex – Carrier disabled, QPSK mode (ignored by modem)
Byte 5 – 00 hex – No Preamble, No Burst(ignored by modem)
Byte 6 – 08 hex – AUPC Off, Mute Automatic, 75 Ohm (ignored by modem)
Byte 7 – 00 hex - Spares
Bytes 8-11 – 60 19 73 04 hex – Frequency = 74.652000 MHz
Bytes 12-13 – 00 00 hex – Used for L-Band
Bytes 14-17 – 00 00 00 00 hex – No Carrier Offset
Bytes 18-19 – 9C FF hex – Minus 10 dBm output level (Shown but ignored)
Bytes 20-21 – 3C 00 hex – AUPC Eb/No setting (00 3C Hex = 6.0 in 0.1dB steps)
Bytes 22-23 – 32 00 hex – AUPC Max Cxr level allowed (00 32 Hex = +5.0 in 0.1dB steps)

Bytes 24-25 – A2 FE hex – AUPC Min Cxr Level. (FE A2 Hex = -35.0 in 0.1dB steps)
Bytes 26-35 – 00 00 00 00 hex – Spare

Packet Ending – "96 C0"

Several things should be noticed here. First, the Write Enable flags do not always cover the enable change for all of the Write bit flags themselves, therefore the Carrier Enable must be repeated from the previous response or set for this specific case. Second, a value for the transmit output level, AUPC levels, etc was used although it was not necessary. This could have been all zeros since the enable flag did not enable setting this parameter.

3. **Read the response from the modem showing the correctly set transmit frequency.**

The response will look the same as the original request command except that the frequency change flag should be set and the data bytes for the frequency should now read "60 19 73 04" hex – meaning the Frequency = 74.652000 MHz

The response status byte is shown as "76" hex, and the error byte was taken as "00". The "76" status byte means Alarm A and B is active, and the Unit, Mod and Demod is in alarm. That was because the modem has no Data or IF connections. These status bits will change depending on other factors in the modem.

Following is an abbreviated list of currently available commands and the parameters monitored and/or controlled by that command. Click on the underlined "Command Byte" to jump to that command table.

Far End Modem Control via the MCC and M5 Compatibility

The M5 and the M500 series protocols contain the ability to relay control packets and responses via the Modem Control Channel, or MCC, from a local to a single far end modem. The PSM-500 modem can interoperate with the previous generation of M5 modem series, the PSM-4900. To aid in mixed systems containing both M500 and M5 series modems the M500 series has almost complete M5 series protocol capabilities. This exists side by side with the newer protocols described in this addendum, but are limited naturally to the capabilities of the M5 series. For further information on the M5 series protocol, please see the M5 Appendix B available on the web site.

As noted before the M5 series of protocols use a "Mode" byte value of "1" to indicate a local packet, and a value of 2 to indicate a far end packet. The M500 series uses a value of "5" for local and "6" for far end packets. Far end packets look just like local packets in all other respects, including the use of the local modem's address. When a modem receives a valid control packet with a mode byte value of 6 (M500) or 2 (M5) then it retransmits that packet to the far end modem via the MCC. You must use the local modem's address in the packet or it will be rejected.

M500 series protocol compatibility with the M5 series extends to the use of the Modem Control Channel to send and receive far end packets. That means that a local M500 series modem can be set up to link to and control either another M500 modem or an M5 series modem at the far end. The M500 series has a built in multiplexer, but the multiplexer is an option in the M5 series. See the respective manuals for information on setting up the link and multiplexers to allow this type of control. The following link packet protocols can be used.

MCC Channel Far End Control Modes			
Controller End Modem	**Far End Modem**	**Mode Byte**	**Possible?**
M500	M500	6	Yes
M500	M500	2	Yes
M500	M5	2	Yes
M5	M500	2	Yes
M5	M500	6	No

The last case is not currently possible because the M5, PSM-4900 series modems do not know how to process an M500 protocol mode byte.

A far end designated packet is not evaluated or processed by the local modem except that its source and destination addresses are removed. When the response is returned by the far end modem, the addresses are inserted into the response before returning it to the original calling controller.

MCC Channel Timing

A packet sent to the far end via the MCC channel may take quite a while before a response is returned to the original caller. The timing also varies depending on the overhead rate assigned to the MCC channel. For example, a 9 byte request with a 50 byte response and a 9600 baud MCC channel overhead would take approximately 9 mS for local receipt, plus 9 mS to send via the MCC, plus 250 mS satellite time, then for the response, approximately 6 mS processing time, 50 mS to send, plus 250 mS satellite return time, plus 50 mS to retransmit to the originator. That is a total of approximately 624 mS. If the MCC overhead rate was lowered to 1200 baud then the time would increase to approximately 1237 mS! The originating controller must be set up to accept this order of delay.

Far End Modem Control via the ESC

The M500, and the M5 series modems equipped with the multiplexer, also have an overhead channel or Engineering Service Channel (ESC), which can be used to relay multiple types of control messages from one station to another. Although it requires more connecting cables, this method has several additional configurations not possible using the MCC.

The ESC is an asynchronous overhead channel capable of being formatted as either RS-232 or RS-485 physically. To use the ESC for control of one or more modems at a far end, the ESC locally is connected to a controlling computer. The ESC at the far end of the link is looped back into the modem's normal control port at J6. To control a single far end modem the physical connection could be either RS-232 or RS-485, but if RS-485 is chosen then multiple modems at a remote location can be controlled by simply daisy chaining the RS-485 to each modem. The ESC at the two ends of the link do not have to be the same physical protocol or even the same data rate since the ESC channel contains buffering at both ends.

Since the ESC channel is simply a link, and knows nothing about the information being carried, it can also be used to control a single far end modem using the VT100 control mode. This mode is limited to a single modem however since that method does not use addresses to point to a particular modem. The VT100 mode has the advantage that all of the programming required is contained within the modem. The controller simply needs to be a dumb terminal or a PC running a terminal emulation program like Hyperterminal in a Windows OS PC.

For information on building the cables needed to connect a computer or a modem's control port to the ESC port connections on the modem's Aux Port (J4), see Appendix C, Cabling, Section 3.

More Information on Modem Monitor and Control

Please see the separate Datum Systems Application Note 18 entitled "*Remote Control of Satellite Modems*". This App Note is available on our web site. It describes several scenarios for control of local and remote modems within a system.

Efficient Monitor and Control Methods

A new set of software for the M500 series of modems released in 2010 beginning with Version 1.19 includes a special "Unit Polling Status" binary packet, command # 0x0A, designed to make monitor and control easier and more efficient. SnIP m500ctl programs at version 0.36 and above coupled with the filesystem 0.6.09 and above can use that software to advantage.

The modem remote binary packets and front panel menus reflect a division of control into 4 functional areas: Unit, Modulator, Demodulator and Interface. These may be logical for human control on the front panel but are not ideal for remote monitor and control. From that perspective modem data might best be divided into 3 categories:

1. **Fixed Data** - Should never change except on power-up. (e.g. Unit Type or Serial Num.)
2. **Configurable Data** - Is changed on command, never by itself (e.g. Mod Data Rate)
3. **Status Notification** - Can change autonomously like Eb/No, Alarms, Tests, etc.

The purpose of this new Unit Polling Status packet is to allow an M&C system like the SnIP itself, or an SNMP agent or client to efficiently manage data and not constantly resend huge volumes of never changing data in order to transfer simple information needed. The structure of the new 0x0A packet allows that. It contains all of the "common" required current status information plus notification of other items which have changed in a fairly concise manner. See the command table for its contents.

The SnIP polling routines and the m500ctl use this one packet to create an internal demand driven database of modem data. This makes most m500ctl functions 2 to 15 times faster, and allows low latency status information to create an efficient AJAX based web that displays status information live (within a few seconds delay)

How it works in the m500ctl program is fairly simple, and similar procedures can be used in an external M&C system connected to the modem's remote control port. The SnIP's background modem polling routine periodically (every 1 to 5 seconds) requests the 0x0A packet data and stores that information in a temporary file. Then as a request for data from a particular packet arrives a decision is made based on **1)** local availability of that packet data plus **2)** the polling status information indicating if that data has changed since the last read (is fresh). If either is not satisfied then the packet with the desired data is read from the modem and the results stored for future use. If a future request can be satisfied from the local data copy then no request is made to the modem.

This system is called “demand driven” because no periodic polling is done of all the data. Instead data is stored locally only when a request demands that data. Only the Polling Status data is periodically requested. The poll timing can also be demand driven, where repeated requests cause the polling interval to decrease.

The Polling Status packet contains virtually all data in category 3 above. Items in category 3 are masked within the modem from the Command Change flags in Bytes 10 to 16 of the Polling Status packet shown below. Thus those change flags indicate that other items in packet status has changed. This keeps you from re-reading a packet for changed data you already have. If however configurable data in category 2 is changed by someone else that bit will be set indicating that the packet needs to be requested again.

The Change flags in Bytes 0 to 3 of each packet and the Status flags are “latched”, that is they hold a change in status until read. Therefore changes are not lost if they return to their previous state between reads. The read interval does not determine if an event is lost, just the granularity of its detection. The Unit Polling Status command has a slightly different set of change flags because of its purpose. The Byte 0-3 change flags are reset upon reading this packet, as in all other packets, but the command change flags in Bytes 10-16 are cleared by reading the particular packet that the flag refers to. A separate set of individual command change flags are maintained by the modem for each interface, the internal one used by the SnIP, the rear panel remote control port and the MCC channel control port.

The Polling Status packet also permits easier implementation of an “event recorder” where significant events are time stamped and stored for later review.

Command Number Index		
Command Byte (hex)	**Name**	**Description**
00	Unit Status	Current Unit Status
01	Unit Config	Unit Configuration
02	Unit Keybrd	Unit Keyboard Setup – Click, Backlight
03	Unit Remote	Unit Remote Control Port Setup – Address, rate, format
04	Unit USB	Unit USB Control Setup
05	Unit Ref	Unit Reference Oscillator Setup – Source and Calibration
06	Unit Redundancy	Unit Redundancy Setup – Enable and monitored alarms
07	Unit Monitor	Unit auxiliary analog voltage monitor output for AGC or Eb/No or Mod CXR Level
08	Unit Alarm	Unit Alarm and Relay Setup
09	Unit Test	Unit Tests – Self Test
0A	Unit Polling	Unit Polling to allow easier determination of changed parameters.
40	Mod Status	Current Modulator Status
41	Mod IF	Transmit IF Control – Frequency, Offset, Level, AUPC, Carrier Enable
42	Mod Data	Modulator Data Setup - Bit rate, FEC
43	Mod Alarm	Modulator Alarm Setup
44	Mod Test	Modulator Test Modes -
45	Mod Mux	IBS Multiplexer Control
46	Mod BUC	Control of Transmit Block Up Converter for PSM-500L
80	Demod Status	Current Demodulator Status
81	Demod IF	Receive IF Control – Frequency, Offset, Acquisition
82	Demod Data	Demodulator Data Setup – Bit Rate, FEC
83	Demod Alarm	Demodulator Alarm Setup
84	Demod Test	Demodulator Test Modes
85	Demod Mux	IBS Multiplexer Control
86	Demod LNB	Control of Receive LNB Down Converter for PSM-500H&L
C0	Intf Status	Interface Status
C1	Intf I/O	Interface Type and Control Line use.
C2	Intf Alarm	Interface Alarm Status
C3	Intf Test	Interface Test Setup – Data Loop-backs, BER Control
C4	Intf SDMS	SDMS Option (Ethernet) Setup/Control – IP Address & Mask
C4	Intf SnIP	SnIP Option (Ethernet) Setup/Control – IP Address & Mask

About the Command Tables:

Each command number begins on a new page. The command number is shown on the first line. There are two main groups of bit/byte sequences included in each command; Read (from Modem) and Write (to Modem).

The “Read” sequence represents the response from a request using that command number and includes “Change” flags, “Read” [Data] flags and “Read” [Data] bytes. All shown information is returned. The “Change” flags show what parameter has changed since the last read, and is reset by the read action. Each control method contains its own set of change flags so that a read from the USB port does not reset the change flags from the serial control port.

The “Write” sequence represents the Command request to set a new parameter(s) using that command number and includes “Enable” flags, “Write” [Data] flags and “Write” [Data] bytes. Only the item(s) with enable flags set are changed. The other values must be present in the packet, but may be “dummy” values since they are not used.

The command or Write sequence to request information only is the command number with the packet data byte count set to zero.

Notes:

1. ***The "Common Notes", Error and Warning status information are at the end of the tables.***
2. ***Table A, which is referred to in the Mod and Demod FEC definitions is at the document end.***
3. ***The bit order is shown lsb on the left to msb on the right. Most programming languages would use a reverse order to denote for example a hexadecimal value. e.g. 0xC4 would be the bits 1100-0100 with the msb on the left and the lsb on the right. Keep this in mind when programming.***

Unit Status, Command [00h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Modem	Ref	Redun	Name	Model	Serial#	Test	Spare
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Status, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	OnLine	McxrEn	DemLck	ModAlm	DemAlm	RefAlm	OcxoAlm	ClkAlm
Byte 5	RdnAlm	NoBck Alm	BckUp Alm	OvrTmp Alm	MHrdFail	DHrdFail	FecAFail	FecBFail
Byte 6	IntfOpt Fail	SndDta Act	RcvDta Act	UnitTst	ModTst	DemTst	IntfTst	RdnEn
Byte 7	MBurst Opt	DBurst Opt	DTpc4 Opt	DTpc16 Opt	DLdpc2 Opt	Spare	Spare	Spare
Byte 8	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 9	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Status, Read Bytes

Bytes 10-26	Unit Name String Terminated with a 00h
Bytes 27-43	Unit Model Number String Terminated with a 00h
Bytes 44-60	Unit Feature Set String Terminated with a 00h
Bytes 61-77	Unit Software Version String Terminated with a 00h
Bytes 78-81	Unit Serial Number, 32b
Bytes 82-83	Fec A Type, 16b, 0=Not Installed
Bytes 84-85	Fec A Version, 16b
Bytes 86-89	Fec A Serial Number, 32b
Bytes 90-91	Fec B Type, 16b, 0=Not Installed
Bytes 92-93	Fec B Version, 16b
Bytes 94-97	Fec B Serial Number, 32b
Bytes 98-99	Interface Option Type, 16b, 0=Not Installed
Bytes 100-101	Interface Option Version, 16b
Bytes 102-105	Interface Option Serial Number, 32b
Bytes 106-127	Spare

Unit Status, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	0	0	RedSW	Name	0	0	0	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Unit Status, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	XferRqst	x	x	x	x	x	x	x
Byte 5	x	x	x	x	x	x	x	x
Byte 6	x	x	x	x	x	x	x	Spare
Byte 7	x	x	x	Spare	Spare	Spare	Spare	Spare

Byte 8	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 9	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Status Write Bytes

Bytes 10-26	Unit Name String Terminated with a 00h
Bytes 27-105	x
Bytes 106-127	Spare

[RdnEn] Flag added in Rev 0.2 to aid in working from a single packet.

FEC Types available are determined by the FEC A and B Version numbers plus the 3 bits DTpc4Opt, DTpc16Opt and DLdpc2Opt. as follows:

- FEC Type 1 = Standard Viterbi plus R-S.
- FEC Type 2 = Standard Viterbi plus R-S and TPC4k hardware.
- FEC Type 3 = Standard Viterbi plus R-S and TPC16k hardware.
- FEC Type 4 = Standard Viterbi plus R-S and TPC4k plus TPC16k hardware.
- FEC Types 1 to 4 can be equipped with LDPC2k Firmware, indicated by the DLdpc2Opt bit set.
- FEC Type 5 is a new universal board only with LDPC16k Firmware by default. The bits DTpc4Opt and DTpc16Opt determine if it is equipped with the TPC4k and/or 16k hardware.

Unit Config, Command [01h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Config	Recall	Store	Restore1	Restore2	Restore3	Restore4	Restore5
Byte 1	Restore6	Restore7	Restore8	PwrUp	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Config, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	ModOnly	ModEn	DemOnly	DemEn	0	0	0	0
Byte 5	0	0	0	0	0	Spare	Spare	Spare
Byte 6	PwrUp0	PwrUp1	PwrUp2	PwrUp3	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Config Read Bytes

Bytes 8-9	Restore 1 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments
Bytes 10-11	Restore 2 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments
Bytes 12-13	Restore 3 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments
Bytes 14-15	Restore 4 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments
Bytes 16-17	Restore 5 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments
Bytes 18-19	Restore 6 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments
Bytes 20-21	Restore 7 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments
Bytes 22-23	Restore 8 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments
Bytes 24-27	Spare

Unit Config, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Config	Recall	Store	Restore1	Restore2	Restore3	Restore4	Restore5
Byte 1	Restore6	Restore7	Restore8	PwrUp	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Unit Config, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	0	ModEn	0	DemEn	Recall0	Recall1	Recall2	Recall3
Byte 5	Store0	Store1	Store2	Store3	x	Spare	Spare	Spare
Byte 6	PwrUp0	PwrUp1	PwrUp2	PwrUp3	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Config Write Bytes

Bytes 8-9	Restore 1 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments
Bytes 10-11	Restore 2 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments
Bytes 12-13	Restore 3 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments
Bytes 14-15	Restore 4 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments
Bytes 16-17	Restore 5 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments
Bytes 18-19	Restore 6 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments
Bytes 20-21	Restore 7 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments
Bytes 22-23	Restore 8 Delay, 16b, (0 to 14,400), 0=Disable Restore, 1 Second Increments

Bytes 24-27	Spare

[Store3-Store0] = Configuration Store, 4b, 0=Factory (Once Only), (1 to 8)=User
[Recall3-Recall0] = Configuration Recall, 4b, 0=Factory, (1 to 8)=User
[PwrUp3- PwrUp 0] = Power Up Configuration Recall, 4b, 0=Last, (1 to 8)=Recall Configuration #.

Unit Keybrd, Command [02h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mode	Entry	LcdActve	LcdIdle	LcdCntst	Activity	Spare	Spare
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Keybrd, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	FpEn	Mode0	Mode1	Entry0	Entry1	Actve0	Actve1	Idle0
Byte 5	Idle1	Actvty0	Actvty1	LcdEn	LcdBad	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Keybrd, Read Bytes

Byte 8	Lcd Contrast, 8b, (0 to 20)
Byte 9	00h
Bytes 10-11	Spare

Unit Keybrd, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mode	Entry	LcdActve	LcdIdle	LcdCntst	Activity	0	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Unit Keybrd, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	x	Mode0	Mode1	Entry0	Entry1	Actve0	Actve1	Idle0
Byte 5	Idle1	Actvty0	Actvty1	x	x	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Keybrd, Write Bytes

Byte 8	Lcd Contrast, 8b, (0 to 20)
Byte 9	00h
Bytes 10-11	Spare

[Mode1-Mode0] = Keyboard Mode, 2b, 0=Disabled, 1=Read Only, 2=Read & Write
[Entry1-Entry0] = Keyboard Entry, 2b, 0=Quick, 1=Edit Only, 2=Confirm
[Actve1-Actve0] = Lcd Active Backlight, 2b, 0=Off, 1=1/3, 2=2/3, 3=Full
[Idle1-Idle0] = Lcd Idle Backlight, 2b, 0=Off, 1=1/3, 2=2/3, 3=Full
[Actvty1-Actvty0] = Key Activity, 2b, 0=None, 1=Beep, 2=Blink, 3=Beep & Blink

Unit Remote, Command [03h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mode	Protocol	Address	Rate	Format	Port	Activity	Spare
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Remote, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mode0	Mode1	Proto0	Proto1	Proto2	Proto3	Rate0	Rate1
Byte 5	Rate2	Rate3	Frmt0	Frmt1	Frmt2	Port	Actvty0	Actvty1
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Remote, Read Bytes

Byte 8	Unit Address, 8b, (0 to 255), 0=None, 255=Global
Byte 9	00h
Bytes 10-11	Spare

Unit Remote, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mode	Protocol	Address	Rate	Format	Port	Activity	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Unit Remote, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mode0	Mode1	Proto0	Proto1	Proto2	Proto3	Rate0	Rate1
Byte 5	Rate2	Rate3	Frmt0	Frmt1	Frmt2	Port	Actvty0	Actvty1
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Remote, Write Bytes

Byte 8	Unit Address, 8b, (0 to 255), 0=None, 255=Global
Byte 9	00h
Bytes 10-11	Spare

[Mode1-Mode0] = Remote Mode, 2b, 0=Disabled, 1=Read Only, 2=Read & Write
[Proto3-Proto0] = Remote Protocol, 4b, 0=VT100, 1=Quiet VT100, 2=Binary Packet A
[Rate3-Rate0] = Remote Rate, 4b, 0=300, 1=600, 2=1200, 3=2400, 4=4800, 5=9600, 6=19200, 7=38400
[Frmt2-Frmt0] = Remote Format, 3b, 0=N81, 1=E81, 2=O81, 3=M81, 4=S81
[Port] = Remote Port, 1b, 0=RS-232, 1=RS-485
[Actvty1-Actvty0] = Remote Activity, 2b, 0=None, 1=Beep, 2=Blink, 3=Beep & Blink

Unit USB, Command [04h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mode	Activity	Spare	Spare	Spare	Spare	Spare	Spare
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit USB, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mode0	Mode1	Actvty0	Actvty1	Spare	Spare	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit USB, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mode	Activity	0	0	0	0	0	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Unit USB, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mode0	Mode1	Actvty0	Actvty1	Spare	Spare	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

[Mode1-Mode0] = USB Mode, 2b, 0=Disabled, 1=Read Only, 2=Read & Write
[Actvty1-Actvty0] = USB Activity, 2b, 0=None, 1=Beep, 2=Blink, 3=Beep & Blink

Unit Ref, Command [05h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Source	Freq.	FineTune	Spare	Spare	Spare	Spare	Spare
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Ref, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Source	Freq0	Freq1	RefCal Error	Spare	Spare	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Ref, Read Bytes

Bytes 6-7	Reference Fine Tune, Signed 16b, (-128 to +127)
Bytes 8-11	Spare

Unit Ref, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Source	Freq.	FineTune	0	0	0	0	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Unit Ref, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Source	Freq0	Freq1	x	Spare	Spare	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Ref, Write Bytes

Bytes 6-7	Reference Fine Tune, Signed 16b, (-128 to +127), Approximately 0.1 PPM per Step
Bytes 8-11	Spare

[Source] = Reference Source, 1b, 0=Internal, 1=External
[Freq1-Freq0] = Reference Frequency, 2b, 0=1.0MHz, 1=5.0MHz, 2=9.0MHz, 3=10.0MHz

Unit Redundancy, Command [06h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mode	SwRqst	SwHold	Config	Spare	Spare	Spare	Spare
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Redundancy, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mode0	Mode1	SwRqst0	SwRqst1	SndCfg	OnLine	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Redundancy, Read Bytes

Bytes 8-9	Switch Hold Time, 16b, (0 to 6,000), 100ms Increments
Bytes 10-11	Spare

Unit Redundancy, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mode	SwRqst	SwHold	Config	0	0	0	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Unit Redundancy, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mode0	Mode1	SwRqst0	SwRqst1	SndCfg	Spare	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Redundancy, Write Bytes

Bytes 8-9	Switch Hold Time, 16b, (0 to 6,000), 100ms Increments
Bytes 10-11	Spare

[Mode1-Mode0] = Redundancy Mode, 2b, 0=Disabled, 1=Internal 1:1, 2=External
[SwRqst1-SwRqst0] = Switch Request, 2b, 0=Any Alarm, 1=Alarm A, 2=Alarm B, 3=Alarm A&B
[SndCfg] = Config Backup, 1b, 0=Idle, 1=Send Config to Backup Unit (This Unit Must be Online)
[OnLine] Flag added in Rev 0.2 to aid in working from a single packet.

Unit Monitor, Command [07h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mode	Full	Zero	Spare	Spare	Spare	Spare	Spare
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Monitor, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mode0	Mode1	Spare	Spare	Spare	Spare	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Monitor, Read Bytes

Bytes 6-7	Full Scale Voltage, Signed 16b, (-100 to +100), 100mV Increments
Bytes 8-9	Zero Scale Voltage, Signed 16b, (-100 to +100), 100mV Increments

Unit Monitor, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mode	Full	Zero	0	0	0	0	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Unit Monitor, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mode0	Mode1	Spare	Spare	Spare	Spare	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Monitor, Write Bytes

Bytes 6-7	Full Scale Voltage, Signed 16b, (-100 to +100), 100mV Increments
Bytes 8-9	Zero Scale Voltage, Signed 16b, (-100 to +100), 100mV Increments

[Mode1-Mode0] = Monitor Mode, 2b, 0=AGC Voltage, 1=Eb/No, 2=Mod CXR Level

Unit Alarm, Command [08h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	RefAlm	OcxoAlm	TstAlm	HardAlm	AlmBeep	Spare	Spare	Spare
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Alarm, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	RefAlm0	RefAlm1	RefAlm2	RefAlm3	OcxAlm0	OcxAlm1	OcxAlm2	OcxAlm3
Byte 5	TstAlm0	TstAlm1	HrdAlm0	HrdAlm1	Beep0	Beep1	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Alarm, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	RefAlm	OcxoAlm	TstAlm	HrdAlm	AlmBeep	0	0	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Unit Alarm, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	RefAlm0	RefAlm1	RefAlm2	RefAlm3	OcxAlm0	OcxAlm1	OcxAlm2	OcxAlm3
Byte 5	TstAlm0	TstAlm1	HrdAlm0	HrdAlm1	Beep0	Beep1	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

[RefAlm3-RefAlm0] = Reference Alarm Mode, 4b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B, 4=Mute CXR, 5=Mute CXR & Alarm A, 6=Mute CXR & Alarm B, 7=Mute CXR & Alarm A&B, 8=Mute BUC & CXR, 9=Mute BUC/CXR & Alarm A, 10=Mute BUC/CXR & Alarm B, 11=Mute BUC/CXR & Alarm A&B
[OcxAlm3-OcxAlm0] = OCXO Alarm Mode, 4b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B, 4=Mute CXR, 5=Mute CXR & Alarm A, 6=Mute CXR & Alarm B, 7=Mute CXR & Alarm A&B, 8=Mute BUC & CXR, 9=Mute BUC/CXR & Alarm A, 10=Mute BUC/CXR & Alarm B, 11=Mute BUC/CXR & Alarm A&B
[TstAlm1-TstAlm0] = Test Active Alarm Mode, 2b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B
[HrdAlm1-HrdAlm0] = Hardware Alarm Mode, 2b, 0=Mute CXR, 1= Mute CXR & Alarm A, 2= Mute CXR & Alarm B, 3= Mute CXR & Alarm A&B
[Beep1-Beep0] = Beeper Alarm Mode, 2b, 0=None, 1=On Alarm A, 2=On Alarm B, 3=On Alarm A&B

Unit Test, Command [09h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Modem	CalRef	UpROM	RefAFC	ClkAFC	+3.3V	+5.0V	+12.0V
Byte 1	+21.0V	-12.0V	BootCode	x	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Test, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Modem0	Modem1	Modem2	Modem3	CalRef	0	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Test, Read Bytes

Bytes 8-9	Ref AFC Voltage, Signed 16b, 100mV Increments
Bytes 10-11	SysClk AFC Voltage, Signed 16b, 100mV Increments
Bytes 12-13	+3.3 Voltage, Signed 16b, 100mV Increments
Bytes 14-15	+5.0 Voltage, Signed 16b, 100mV Increments
Bytes 16-17	+12.0 Voltage, Signed 16b, 100mV Increments
Bytes 18-19	+21.0 Voltage, Signed 16b, 100mV Increments
Bytes 20-21	-12.0 Voltage, Signed 16b, 100mV Increments
Bytes 22-23	Boot Error Code, 16b
Bytes 24-25	Boot Error Page Address, 16b
Bytes 26-27	Boot Error Segment Address, 16b
Byte 28	Self Test Step Status, 8b
Bytes 29-33	Self Test Status, 56b
Bytes 34-35	Reference Calibration Status, 16b
Bytes 36-41	Spare

Unit Test, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Modem	CalRef	UpROM	0	0	0	0	0
Byte 1	0	0	BootStat	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Unit Test, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Modem0	Modem1	Modem2	Modem3	CalRef	BootStat	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Unit Test, Write Bytes

Bytes 8-11	*Confirm Serial Number, 32b, Required to Enable Flash ROM Update
Bytes 12-13	x
Bytes 14-15	x
Bytes 16-17	x

Bytes 18-19	x
Bytes 20-21	x
Bytes 22-23	x
Bytes 24-25	x
Bytes 26-27	x
Bytes 28-33	x
Bytes 34-35	x
Bytes 36-41	Spare

*****If **UpROM** Enabled then Serial Number Confirmation Required else don't care
[Modem3-Modem0] = Modem Self Test, 4b, 0=Normal, 1=Lamp Test, 2=Self Test, 3=Lamp & Self Test
[CalRef] = Reference Calibration, 1b, 0=Normal, 1=Enabled
[BootStat] = CPU Boot Status Code, 1b, 1=Reset Boot Code to Zero

Self Test Step Status

[Byte 28] Value: 1=Self Test Initializing, 2=Alarm On Test, 3=Upper Limit Test, 4=Lower Limit Test, 5=Loop Test 1 (4.92Mbps), 6=Loop Test 2 (2.4kbps), 100=Self Test Completed Successfully.

Self Test Status

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 29	DHrdFail	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Byte 30	SysAlm Bad	MStpAlm Bad	MLoAlm Bad	DStpAlm Bad	DLoAlm Bad	MBitAlm Bad	3.3VAlm	5.0VAlm
Byte 31	12VAlm	21Valm	-12VAlm	SysAlm	MStpAlm	MLoAlm	DStpAlm	DLoAlm
Byte 32	MBitAlm	MlvlAlm	DIQAlm	OvrTAlm	SysAfc Alm	MStpAfc Alm	MLoAfc Alm	DStpAfc Alm
Byte 33	DLoAfc Alm	Lock Failed	EbNo Error	Offset Error	Level Error	Lock Error	SyncLoss Error	BER Error

Reference Calibration Status

[Bytes 34-35] Value: 1=Bad Input Reference, 2=DAC Error, 3=Flash Write Error, 4=Calibration Error, 5=Reference Calibration Completed Successfully.

Unit Polling Status, Command [0Ah]

Read Change Flags [Note 1]

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	UnitStat	UnitCfg	ModStat	ModCfg	DemStat	Demcfg	IntStat	IntCfg
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Front Panel LED Image, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	ModCxr	ModMjr	ModMnr	ModTest	DemLock	DemMjr	DemMnr	DemTest
Byte 5	UnitSum	UnitLocal	UnitRemt	Spare	Spare	Spare	Spare	Spare

Modem Status and Alarm Flags, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 6	OnLine	MCxrEn	DemLck	ModAlm	DemAlm	RefAlm	OcxoAlm	ClkAlm
Byte 7	RdnAlm	NoBckAlm	BckUpAlm	TempAlm	MHrdFail	DHrdFail	FecAFail	FecBFail
Byte 8	IntOptFail	SndDtaAct	RcvDtaAct	UnitTst	ModTst	DmdTst	IntfTst	RdnEn
Byte 9	BufStat	BufSlip	BufSlipSign	BucAlm	LnbAlm			

Command Change Flags, Read Flags [Note 2]

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 10	Cmd 00	Cmd 01	Cmd 02	Cmd 03	Cmd 04	Cmd 05	Cmd 06	Cmd 07
Byte 11	Cmd 08	Cmd 09	Cmd 0A					
Byte 12	Cmd 40	Cmd 41	Cmd 42	Cmd 43	Cmd 44	Cmd 45	Cmd 46	
Byte 13								
Byte 14	Cmd 80	Cmd 81	Cmd 82	Cmd 83	Cmd 84	Cmd 85	Cmd 86	
Byte 15								
Byte 16	Cmd C0	Cmd C1	Cmd C2	Cmd C3	Cmd C4			

Unit Volatile Data, Read Bytes

Bytes 17	Buffer Fill Status, UnSigned 8b, (0 to +200), 1% Increments
Bytes 18-19	Receive Eb/No, UnSigned 16b, (0 to 200), 0.1dB Increments
Bytes 20-23	Receive Carrier Offset, Signed 32b, 1Hz Increments
Bytes 24-25	Receive Carrier Level, Signed 16b, 0.1dB Increments
Bytes 26-27	Receive Estimated BER, UnSigned 16b, (0 to 200), 0.1dB Increments
Bytes 28-29	Transmit Carrier Level, Signed 16b, (-35 to +5), 0.1dB Increments
Bytes 30-31	Spare

Notes:

1. **The Read Change Flags are cleared by reading this command.**
2. **The Command Change Flags are cleared by reading their respective command.**
3. **All data is read only.**

This command is intended to aid in minimizing constant polling of all commands to determine status. All data that changes autonomously is contained within this response. The command change flags can be used to determine if a parameter is changed by another source, for example via the front panel. It can also be used to help create a local database of modem parameters.

Mod Status, Command [40h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	CXR	Data	Clock	Test	Spare	Spare	Spare	Spare
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod Status, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Online	CxrEn	CXRAlm	RTSMute	DLckAlm	APCLmt	APCAlm	DataTmg Alm
Byte 5	DtaAct Alm	Clock Alm	LvlAlm	LoAlm	StpAlm	SysAlm	RefAlm	OcxoAlm
Byte 6	FecAlm	PureCXR	Alt1/0	Sideband	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod Status, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	0	0	0	0	0	0	0	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Mod Status, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	x	x	x	x	x	x	x	x
Byte 5	x	x	x	x	x	x	x	x
Byte 6	x	x	x	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod IF, Command [41h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Freq	Offset	Level	Output	Mod	Spectrum	Filter	Mode
Byte 1	Preamble	AUPC	AEbNo	AMaxLvl	AMinLvl	Mute	Imped	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod IF, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	CxrEn	Mod0	Mod1	Mod2	Mod3	SpcInv	Filter0	Filter1
Byte 5	Mode0	Mode1	Mode2	Mode3	PreLng0	PreLng1	PreLng2	PreLng3
Byte 6	AUPCEn	Mute0	Mute1	ImpSel	CxrHrd	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod IF, Read Bytes

Bytes 8-11	CXR Frequency, 32b, (50,000,000 to 90,000,000 for 70 MHz units, 100,000,000 to 180,000,000 for 140 MHz units or 950,000,000 to 1,750,000,000 MHz for L-Band units if the BUC LO frequency = 0), 1Hz Increments
Bytes 12-13	CXR Frequency, Reserved for L-Band Frequency Extension when BUC LO not = 0. Note 1
Bytes 14-17	CXR Offset, Signed 32b, (-1,250,000 to +1,250,000), 1Hz Increments
Bytes 18-19	CXR Level, Signed 16b, (-350 to +50*), 0.1dB Increments (dBm)
Bytes 20-21	AUPC Eb/No, 16b, (30 to 200), 0.1dB Increments
Bytes 22-23	AUPC Max Level, Signed 16b, (-350 to +50*), 0.1dB Increments
Bytes 24-25	AUPC Min Level, Signed 16b, (-350 to +50*), 0.1dB Increments
Bytes 26-35	Spare

Mod IF, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Freq	Offset	Level	Output	Mod	Spectrum	Filter	Mode
Byte 1	Preamble	AUPC	AEbNo	AmaxLvl	AMinLvl	Mute	Imped	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Mod IF, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	CxrEn	Mod0	Mod1	Mod2	Mod3	SpcInv	Filter0	Filter1
Byte 5	Mode0	Mode1	Mode2	Mode3	PreLng0	PreLng1	PreLng2	PreLng3
Byte 6	AUPCEn	Mute0	Mute1	ImpSel	x	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod IF, Write Bytes

Bytes 8-11	Cxr Frequency, 32b, (50,000,000 to 90,000,000), 1Hz Increments
Bytes 12-13	Cxr Frequency, Reserved for L-Band Frequency Extension
Bytes 14-17	Cxr Offset, Signed 32b, (-1,250,000 to +1,250,000), 1Hz Increments
Bytes 18-19	Cxr Level, Signed 16b, (-350 to +50*), 0.1dB Increments (dBm)
Bytes 20-21	AUPC Eb/No, 16b, (30 to 200), 0.1dB Increments
Bytes 22-23	AUPC Max Level, Signed 16b, (-350 to +50*), 0.1dB Increments
Bytes 24-25	AUPC Min Level, Signed 16b, (-350 to +50*), 0.1dB Increments
Bytes 26-35	Spare

***** +30 (+3.0dBm) When 50 Ohms Output Impedance Selected.

[CxrEn] = Mod Cxr Output Enable, 1b, 0=Disabled, 1=Enabled
[Mod3-Mod0] = Mod Modulation mode, 4b, 0=BPSK, 1=QPSK, 2=OQPSK, 3=8PSK, 4=8QAM, 6=16QAM
[SpcInv] = Mod Spectrum Invert, 1b, 0=Normal, 1=Inverted
[Filter1-Filter0] = Mod Filter Mask, 2b, 0=IESS, 1=Legacy
[Mode3-Mode0] = Mod Burst Mode, 4b, 0=Normal
[PreLng3-PreLng0] = Mod Burst Preamble Length, 4b, Undefined
[AUPCEn] = Mod AUPC Mode, 1b, 0=Disabled, 1=Enabled
[Mute1-Mute0] = Mod Cxr Mute Mode, 2b, 0=Automatic, 1=Confirm, 2=Manual
[ImpSel] = Mod Output Impedance, 1b, 0=50 Ohms, 1=75 Ohms

Note 1. IF/RF Frequency. The full maximum IF frequency range of either 70 MHz, 140 MHz or L-Band units can fit into a standard 32 bit unsigned integer definition. The frequency may exceed that if the BUC LO frequency is set, which caused the modem to calculate and display the RF frequency in this place for operator convenience. The RF frequencies are typically in the 3 to 15 GHz range, requiring more than 32 bits to hold the calculated value. The 2 byte extension shown makes it a 48 bit number that can easily hold the full range of values. Most computers however have either 32 or 64 bit variable sizes, and this 48 bit value will have to be mapped into a 64 bit variable.

Mod Data, Command [42h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	BitRate	Mod	FecType	FecOpt	CodeRate	RsMode	RsN	RsK
Byte 1	RsDepth	DiffEnc	Scrmbler	ClkSrc	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod Data, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mod0	Mod1	Mod2	Mod3	RsMode0	RsMode1	RsMode2	RsMode3
Byte 5	RsDepth0	RsDepth1	DiffEnc0	DifEnc1	Scrm0	Scrm1	Scrm2	Scrm3
Byte 6	ClkSrc0	ClkSrc1	ClkSrc2	ClkSrc3	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod Data, Read Bytes

Bytes 8-11	Bit Rate, 32b, (600 to 20,000,000), 1bps Increments (Depends on Mode)
Byte 12	Fec Type, 8b, 0=None, 1=Viterbi, 2=TCM, 4=TPC, **See Table A**
Byte 13	
Bytes 14-15	Fec Option, 16b, **See Table A**
Bytes 16-17	Fec Code Rate, 16b**, See Table A**
Byte 18	Reed-Solomon N Factor (22 to 255, N-K=2 to 20)
Byte 19	Reed-Solomon K Factor (20 to 253, N-K=2 to 20)
Bytes 20-31	Spare

Mod Data, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	BitRate	Mod	FecType	FecOpt	CodeRate	RsMode	RsN	RsK
Byte 1	RsDepth	0	Scrmbler	ClkSrc	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Mod Data, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mod0	Mod1	Mod2	Mod3	RsMode0	RsMode1	RsMode2	RsMode3
Byte 5	RsDepth0	RsDepth1	x	x	Scrm0	Scrm1	Scrm2	Scrm3
Byte 6	ClkSrc0	ClkSrc1	ClkSrc2	ClkSrc3	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod Data, Write Bytes

Bytes 8-11	Bit Rate, 32b, (600 to 20,000,000), 1bps Increments (Depends on Mode)
Byte 12	Fec Type, 8b, 0=None, 1=Viterbi, 2=TCM, 4=TPC, **See Table A**
Byte 13	x
Bytes 14-15	Fec Option, 16b, **See Table A**
Bytes 16-17	Fec Code Rate, 16b**, See Table A**
Byte 18	Reed-Solomon N Factor (22 to 255, N-K=2 to 20)
Byte 19	Reed-Solomon K Factor (20 to 253, N-K=2 to 20)
Bytes 20-31	Spare

[Mod3-Mod0] = Mod Modulation Mode, 4b, 0=BPSK, 1=QPSK, 2=OQPSK, 3=8PSK, 4=8QAM, 6=16QAM

[RsMode3-RsMode0] = Mod Reed-Solomon Mode, 4b, 0=Disabled, 1=IESS308, 2=IESS309, 3=IESS310, 4=Custom, 5=CT220,200
[RsDepth1-RsDepth0] = Mod Reed-Solomon Interleaver Depth, 2b, 0=4, 1=8, 2=16
[DifEnc1-DifEnc0] = Mod Diff Encoder, 2b, 0=Disabled, 1=Enabled, 2=Symbol **(Read Only)**
[Scrm3-Scrm0] = Mod Scrambler, 4b, 0=Disabled, 1=Auto, 2=V.35, 3=Intelsat, 4=Alt V.35, 5=Alt Intelsat, 6=EFD, 7=RS Sync, 8=IBS Sync, 9=FEC Sync
[ClkSrc3-ClkSrc0] = Mod Clock Source, 4b, 0=Internal, 1=Terminal Timing, 2=External, 3=RCV

Mod Alarm, Command [43h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	CxrAlm	DtaAlm	ClkAlm	ApcAlm	TstAlm	HrdAlm	BucAlm	Spare
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod Alarm, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	CxrAlm0	CxrAlm1	DtaAlm0	DtaAlm1	ClkAlm0	ClkAlm1	ClkAlm2	ClkAlm3
Byte 5	ApcAlm0	ApcAlm1	TstAlm0	TstAlm1	HrdAlm0	HrdAlm1	BucAlm0	BucAlm1
Byte 6	BucAlm2	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod Alarm, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	CxrAlm	DataAlm	ClkAlm	ApcAlm	TstAlm	HrdAlm	BucAlm	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Mod Alarm, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	CxrAlm0	CxrAlm1	DtaAlm0	DtaAlm1	ClkAlm0	ClkAlm1	ClkAlm2	ClkAlm3
Byte 5	ApcAlm0	ApcAlm1	TstAlm0	TstAlm1	HrdAlm0	HrdAlm1	BucAlm0	BucAlm1
Byte 6	BucAlm2	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

[CxrAlm1-CxrAlm0] = Mod Cxr Alarm Mode, 2b, 0=Mute Cxr, 1= Mute Cxr & Alarm A, 2= Mute Cxr & Alarm B, 3= Mute Cxr & Alarm A&B
[DtaAlm1-DtaAlm0] = Mod Data Alarm Mode, 2b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B
[ClkAlm3-ClkAlm0] = Mod Clock Alarm Mode, 4b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B, 4=Send A1S, 5=Send A1S & Alarm A, 6=Send A1S & Alarm B, 7=Send A1S & Alarm A&B, 8=Mute Cxr, 9=Mute Cxr & Alarm A, 10=Mute Cxr & Alarm B, 11=Mute Cxr & Alarm A&B
[ApcAlm1-ApcAlm0] = Mod AUPC Alarm Mode, 2b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B
[TstAlm1-TstAlm0] = Mod Test Active Alarm Mode, 2b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B
[HrdAlm1-HrdAlm0] = Mod Hardware Alarm Mode, 2b, 0=Mute CXR, 1= Mute CXR & Alarm A, 2= Mute CXR & Alarm B, 3= Mute CXR & Alarm A&B
[BucAlm2-BucAlm0] = Mod BUC Power Alarm Mode (L-Band Only), 3b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B, 4=Mute Cxr, 5=Mute Cxr & Alarm A, 6=Mute Cxr & Alarm B, 7=Mute Cxr & Alarm A&B

Mod Test, Command [44h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	TstMod	SymRate	CxrALC	LoAFC	StpAFC	x	x	x
Byte 1	x	x	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod Test, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	TstMod0	TstMod1	Spare	Spare	Spare	Spare	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod Test, Read Bytes

Bytes 6-9	Symbol Rate, 32b, 1 Hz Increments
Bytes 10-11	Cxr ALC Voltage, Signed 16b, 100mV Increments
Bytes 12-13	LO AFC Voltage, Signed 16b, 100mV Increments
Bytes 14-15	Step AFC Voltage, Signed 16b, 100mV Increments
Bytes 16-25	(Reserved) All Zeros
Bytes 26-31	Spare

Mod Test, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	TstMod	0	0	0	0	0	0	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Mod Test, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	TstMod0	TstMod1	Spare	Spare	Spare	Spare	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod Test, Write Bytes

Bytes 6-9	x
Bytes 10-11	x
Bytes 12-13	x
Bytes 14-15	x
Bytes 16-25	x
Bytes 26-31	Spare

[TstMod1-TstMod0] = Mod Test Modulation, 2b, 0=Normal, 1=Pure Cxr, 2=Alt 1/0, 3=Sideband

Mod Mux, Command [45h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mode	Esc Ovrhead	Mcc Ovrhead	Ratio	EscPort	EscRate	Esc Format	EscCts
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod Mux, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mode0	Mode1	Mode2	Mode3	EscOh0	EscOh1	EscOh2	EscOh3
Byte 5	MccOh0	MccOh1	MccOh2	MccOh3	Port0	Port1	Port2	Port3
Byte 6	Rate0	Rate1	Rate2	Rate3	Frmt0	Frmt1	CtsMd	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 8	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 9	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 10	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 11	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod Mux, Read Bytes

Bytes 12-13	Mux Ratio X, Unsigned 16b, (1 to 255), X:Y
Bytes 14-15	Mux Ratio Y, Unsigned 16b, (2 to 256), X:Y
Bytes 16-23	Spare

Mod Mux, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mux Mode	Esc Ovrhead	Mcc Ovrhead	0	EscPort	EscRate	Esc Format	EscCts
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Mod Mux, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mode0	Mode1	Mode2	Mode3	EscOh0	EscOh1	EscOh2	EscOh3
Byte 5	MccOh0	MccOh1	MccOh2	MccOh3	Port0	Port1	Port2	Port3
Byte 6	Rate0	Rate1	Rate2	Rate3	Frmt0	Frmt1	CtsMd	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 8	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 9	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 10	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 11	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod Mux, Write Bytes

Bytes 12-13	x
Bytes 14-15	x
Bytes 16-23	Spare

[Mode3-Mode0] = Mode, 4b, 0=Disabled, 1=Standard IBS, 2=Enhanced IBS, 3=Custom IBS

[EscOh3-EscOh0] = ESC Overhead (Custom Mode Only), 4b, 0=Disabled, 1=300, 2=600, 3=1200, 4=2400, 5=4800, 6=9600, 7=19200, 8=38400
[MccOh3-MccOh0] = MCC Overhead (Custom Mode Only), 4b, 0=Disabled, 1=300, 2=600, 3=1200, 4=2400, 5=4800, 6=9600, 7=19200, 8=38400
[Port3-Port0] = ESC Port, 4b, 0=RS-232, 1=RS-485 2-Wire, 2=RS-485 4-Wire, 3=RS-485 4-Wire Driver On
[Rate3-Rate0] = ESC Rate, 4b, 0=300, 1=600, 2=1200, 3=2400, 4=4800, 5=9600, 6=19200, 7=38400
[Frmt1-Frmt0] = ESC Format, 2b, 0=N71, 1=P71, 2=N81, 3=P81
[CtsMd] = ESC CTS Mode, 1b, 0=Normal (Xmt Flow Control), 1=Ignore

Mod BUC, Command [46h]
Read Change Flags (L-Band Only)

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	BucPwr	VOut	VMin	IOut	IMax	IMin	Ref	LoFrq
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod BUC, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	PwrEn0	PwrEn1	PwrEn2	RefEn0	RefEn1	RefEn2	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod BUC, Read Bytes

Bytes 8-9	BUC Voltage Out, Signed 16b, (0 to >600), 100mV Increments
Bytes 10-11	BUC Voltage Min, Signed 16b, (80 to 600), 100mV Increments
Bytes 12-13	BUC Current Out, Signed 16b, (0 to >600), 10mA Increments
Bytes 14-15	BUC Current Max, Signed 16b, (5 to 600), 10mA Increments
Bytes 16-17	BUC Current Min, Signed 16b, (5 to 600), 10mA Increments
Bytes 18-23	BUC LO Frequency, Unsigned 48b, (0 to 50,000,000,000), 1Hz Increments
Bytes 24-33	Spare

Mod BUC, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	BucPwr	0	VMin	0	IMax	IMin	Ref	LoFrq
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Mod BUC, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	PwrEn0	PwrEn1	PwrEn2	RefEn0	RefEn1	RefEn2	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Mod BUC, Write Bytes

Bytes 8-9	x
Bytes 10-11	BUC Voltage Min, Signed 16b, (80 to 600), 100mV Increments
Bytes 12-13	x
Bytes 14-15	BUC Current Max, Signed 16b, (5 to 600), 10mA Increments
Bytes 16-17	BUC Current Min, Signed 16b, (5 to 600), 10mA Increments
Bytes 18-23	BUC LO Frequency, Unsigned 48b, (0 to 50,000,000,000), 1Hz Increments**. Note 1**
Bytes 24-33	Spare

[PwrEn2-PwrEn0] = BUC Power, 3b, 0=Disabled, 1=Enabled
[RefEn2-RefEn0] = BUC 10MHz Ref, 3b, 0=Disabled, 1=Enabled
Note 1: When the BUC LO frequency is not zero the Mod IF frequency is converted to an RF frequency.

Demod Status, Command [80h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	CXR	Eb/No	Offset	Level	EstBER	SER	Buffer	Test
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod Status, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	DemLck	EbAlm	LvlAlm	AgcAlm	LoAlm	StpAlm	SysAlm	RefAlm
Byte 5	BufValid	BufSlip	BufSlip Sign	NrwHld	DataAlm	BckAlm	IfLoop	FixCxr
Byte 6	LnbAlm	IfLpSyn Alm	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod Status, Read Bytes

Bytes 8-9	Rcv Eb/No, 16b, (20 to 200), 0.1dB Increments
Bytes 10-13	Rcv Offset, Signed 32b, (-1,250,000 to +1,250,000), 1Hz Increments
Bytes 14-15	Rcv Cxr Level, Signed 16b, (<-84 to 0), 1dB Increments (dBm)
Bytes 16-17	Est. BER, 16b, Bits [15-12]=Negative Exponent, Bits [11-0]=Mantissa (1-9, >9=<1)
Bytes 18-19	SER, 16b, Bits [15-12]=Negative Exponent, Bits [11-0]=Mantissa (100=1.00)
Bytes 20-21	Buffer, 16b, (0 to 200), 1% Increments
Bytes 22-27	Spare

Demod Status, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	CxrLck	0	SwpStrt	0	EstBER	SER	Buffer	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Demod Status, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	AbortLck	x	x	x	x	x	x	x
Byte 5	x	CtrBuf	x	x	x	x	x	x
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod Status, Write Bytes

Bytes 8-11	x
Bytes 12-13	Sweep Start Frequency, Signed 32b, (-1,250,000 to +1,250,000), 1Hz Increments
Bytes 14-15	x
Bytes 16-17	x, Write Restarts SER/Est. BER
Bytes 18-19	x, Write Restarts SER/Est. BER
Bytes 20-21	x, Write Clears Slip Flag
Bytes 22-27	Spare

[AbortLck] = Demod Abort Current Lock, 1b, 0=No Change, 1=Abort Lock
[CtrBuf] = Demod Recenter Buffer, 1b, 0=Clear Slip Flag Only, 1=Clear Slip Flag & Recenter Buffer

Demod IF, Command [81h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Freq	SwpRng	SwpMd	SwpTm	Mod	Spectrum	Filter	Mode
Byte 1	Preamble	LowEb	LowLvl	Imped	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod IF, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	SwpMd	Mod0	Mod1	Mod2	Mod3	SpcInv	Filter0	Filter1
Byte 5	Mode0	Mode1	Mode2	Mode3	PreLng0	PreLng1	PreLng2	PreLng3
Byte 6	ImpSel	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod IF, Read Bytes

Bytes 8-11	CXR Frequency, 32b, (50,000,000 to 90,000,000 for 70 MHz units, 100,000,000 to 180,000,000 for 140 MHz units or 950,000,000 to 1,900,000,000 MHz for L-Band units if the LNB LO frequency = 0), 1Hz Increments
Bytes 12-13	CXR Frequency, Reserved for L-Band Frequency Extension if LNB LO not = 0. Note 1
Bytes 14-17	Sweep Range, 32b, (100 to 1,250,000), 1Hz Increments
Bytes 18-19	Sweep Time, 16b, (0 to 6,000), 100ms Increments, 0=Narrow Disabled
Bytes 20-21	Low Eb/No, 16b, (10 to 200), 0.1dB Increments
Bytes 22-23	Low Level, Signed 16b, (-850 to –80), 0.1dB Increments (Dependent on Symbol Rate)
Bytes 24-37	Spare

Demod IF, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Freq	SwpRng	SwpMd	SwpTm	Mod	Spectrum	Filter	Mode
Byte 1	Preamble	LowEb	LowLvl	Imped	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Demod IF, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	SwpMd	Mod0	Mod1	Mod2	Mod3	SpcInv	Filter0	Filter1
Byte 5	Mode0	Mode1	Mode2	Mode3	PreLng0	PreLng1	PreLng2	PreLng3
Byte 6	ImpSel	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod IF, Write Bytes

Bytes 8-11	CXR Frequency, 32b, (50,000,000 to 90,000,000), 1Hz Increments
Bytes 12-13	CXR Frequency, Reserved for L-Band Frequency Extension
Bytes 14-17	Sweep Range, 32b, (100 to 1,250,000), 1Hz Increments
Bytes 18-19	Sweep Time, 16b, (0 to 6,000), 100ms Increments, 0=Narrow Disabled
Bytes 20-21	Low Eb/No, 16b, (10 to 200), 0.1dB Increments
Bytes 22-23	Low Level, Signed 16b, (-850 to –80), 0.1dB Increments (Dependent on Symbol Rate)
Bytes 24-37	Spare

[SwpMd] = Demod Sweep Mode, 1b, 0=Normal, 1=Search

[Mod3-Mod0] = Demod Modulation mode, 4b, 0=BPSK, 1=QPSK, 2=OQPSK, 3=8PSK, 4=8QAM, 6=16QAM
[SpcInv] = Demod Spectrum Invert, 1b, 0=Normal, 1=Inverted
[Filter1-Filter0] = Demod Filter Mask, 2b, 0=IESS, 1=Legacy
[Mode3-Mode0] = Demod Burst Mode, 4b, 0=Normal
[PreLng3-PreLng0] = Demod Burst Preamble Length, 4b, Undefined
[ImpSel] = Demod Input Impedance, 1b, 0=50 Ohms, 1=75 Ohms

Note 1. IF/RF Frequency. The full maximum IF frequency range of either 70 MHz, 140 MHz or L-Band units can fit into a standard 32 bit unsigned integer definition. The frequency may exceed that if the LNB LO frequency is set, which caused the modem to calculate and display the RF frequency in this place for operator convenience. The RF frequencies are typically in the 3 to 15 GHz range, requiring more than 32 bits to hold the calculated value. The 2 byte extension shown makes it a 48 bit number that can easily hold the full range of values. Most computers however have either 32 or 64 bit variable sizes, and this 48 bit value will have to be mapped into a 64 bit variable.

Demod Data, Command [82h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	BitRate	Mod	FecType	FecOpt	CodeRate	RsMode	RsN	RsK
Byte 1	RsDepth	DiffEnc	Scrmbler	ClkSrc	BufDly	BufSize	FecHold	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod Data, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mod0	Mod1	Mod2	Mod3	RsMode0	RsMode1	RsMode2	RsMode3
Byte 5	RsDepth0	RsDepth1	DiffEnc0	DiffEnc1	Scrm0	Scrm1	Scrm2	Scrm3
Byte 6	ClkSrc0	ClkSrc1	ClkSrc2	ClkSrc3	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod Data, Read Bytes

Bytes 8-11	Bit Rate, 32b, (600 to 20,000,000), 1bps Increments (Depends on Mode)
Byte 12	Fec Type, 8b, 0=None, 1=Viterbi, 2=TCM, 4=TPC, **See Table A**
Byte 13	0
Bytes 14-15	Fec Option, 16b, **See Table A**
Bytes 16-17	Fec Code Rate, 16b**, See Table A**
Byte 18	Reed-Solomon N Factor (22 to 255, N-K=2 to 20)
Byte 19	Reed-Solomon K Factor (20 to 253, N-K=2 to 20)
Bytes 20-23	Buffer Delay, 32b, (8 to >266,000), 100ns Increments (Depends on Bit Rate)
Bytes 24-27	Buffer Size, 32b, (4 to 524,284), 1 Bit Increments
Bytes 28-29	Fec Hold Count, 16b, (0 to 255)
Bytes 30-41	Spare

Demod Data, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	BitRate	Mod	FecType	FecOpt	CodeRate	RsMode	RsN	RsK
Byte 1	RsDepth	0	Scrmbler	ClkSrc	BufDly	BufSize	FecHold	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Demod Data, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mod0	Mod1	Mod2	Mod3	RsMode0	RsMode1	RsMode2	RsMode3
Byte 5	RsDepth0	RsDepth1	DiffEnc0	DiffEnc1	Scrm0	Scrm1	Scrm2	Scrm3
Byte 6	ClkSrc0	ClkSrc1	ClkSrc2	ClkSrc3	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod Data, Write Bytes

Bytes 8-11	Bit Rate, 32b, (600 to 20,000,000), 1bps Increments (Depends on Mode)
Byte 12	Fec Type, 8b, 0=None, 1=Viterbi, 2=TCM, 4=TPC, **See Table A**
Byte 13	x
Bytes 14-15	Fec Option, 16b, **See Table A**
Bytes 16-17	Fec Code Rate, 16b**, See Table A**
Byte 18	Reed-Solomon N Factor (22 to 255, N-K=2 to 20)
Byte 19	Reed-Solomon K Factor (20 to 253, N-K=2 to 20)
Bytes 20-23	Buffer Delay, 32b, (8 to >266,000), 100ns Increments (Depends on Bit Rate)

Bytes 24-27	Buffer Size, 32b, (4 to 524,284), 1 Bit Increments
Bytes 28-29	Fec Hold Count, 16b, (0 to 255)
Bytes 30-41	Spare

[Mod3-Mod0] = Demod Modulation Mode, 4b, 0=BPSK, 1=QPSK, 2=OQPSK, 3=8PSK, 4=8QAM, 6=16QAM
[RsMode3-RsMode0] = Demod Reed-Solomon Mode, 4b, 0=Disabled, 1=IESS308, 2=IESS309, 3=IESS310, 4=Custom, 5=CT220,200
[RsDepth1-RsDepth0] = Demod Reed-Solomon Interleaver Depth, 2b, 0=4, 1=8, 2=16
[DifEnc1-DifEnc0] = Demod Diff Encoder, 2b, 0=Disabled, 1=Enabled, 2=Symbol **(Read Only)**
[Scrm3-Scrm0] = Demod Scrambler, 4b, 0=Disabled, 1=Auto, 2=V.35, 3=Intelsat, 4=Alt V.35, 5=Alt Intelsat, 6=EFD, 7=RS Sync, 8=IBS Sync, 9=FEC Sync
[ClkSrc3-ClkSrc0] = Demod Clock Source, 4b, 0=Rcv, 1=Internal, 2=External, 3=Mod Clock

Demod Alarm, Command [83h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	CxrAlm	DtaAlm	EbAlm	LvlAlm	TstAlm	HrdAlm	BckAlm	LnbAlm
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod Alarm, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	CxrAlm0	CxrAlm1	CxrAlm2	DtaAlm0	DtaAlm1	EbAlm0	EbAlm1	LvlAlm0
Byte 5	LvlAlm1	TstAlm0	TstAlm1	HrdAlm0	HrdAlm1	BckAlm0	BckAlm1	LnbAlm0
Byte 6	LnbAlm1	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod Alarm, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	CxrAlm	DtaAlm	EbAlm	LvlAlm	TstAlm	HrdAlm	BckAlm	LnbAlm
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Demod Alarm, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	CxrAlm0	CxrAlm1	CxrAlm2	DtaAlm0	DtaAlm1	EbAlm0	EbAlm1	LvlAlm0
Byte 5	LvlAlm1	TstAlm0	TstAlm1	HrdAlm0	HrdAlm1	BckAlm0	BckAlm1	LnbAlm0
Byte 6	LnbAlm1	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

[CxrAlm2-CxrAlm0] = Demod Cxr Lock Alarm Mode, 3b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B, 4=Mute CXR, 5=Mute CXR & Alarm A, 6=Mute CXR & Alarm B, 7=Mute CXR & Alarm A&B
[EbAlm1-EbAlm0] = Demod Eb/No Alarm Mode, 2b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B
[LvlAlm1-LvlAlm0] = Demod Level Alarm Mode, 2b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B
[TstAlm1-TstAlm0] = Demod Test Active Alarm Mode, 2b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B
[HrdAlm1-HrdAlm0] = Demod Hardware Alarm Mode, 2b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B
[BckAlm1-BckAlm0] = Demod Backward Alarm Mode, 2b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B
[LnbAlm1-LnbAlm0] = Demod LNB Power Alarm Mode, 2b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B

Demod Test, Command [84h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	IfLoop	SymRate	x	AGC	LoAFC	StpAFC	IdcOff	QdcOff
Byte 1	x	x	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod Test, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	IfLoop0	IfLoop1	0	0	Spare	Spare	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod Test, Read Bytes

Bytes 6-9	Symbol Rate, 32b, 1 Hz Increments
Bytes 10-11	AGC Voltage, Signed 16b, 100mV Increments
Bytes 12-13	LO AFC Voltage, Signed 16b, 100mV Increments
Bytes 14-15	Step AFC Voltage, Signed 16b, 100mV Increments
Bytes 16-17	IDcOff Voltage, Signed 16b, 100mV Increments
Bytes 18-19	QDcOff Voltage, Signed 16b, 100mV Increments
Bytes 20-21	(Reserved) All Zeros
Bytes 22-23	(Reserved) All Zeros
Bytes 24-33	Spare

Demod Test, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	IfLoop	0	0	0	0	0	0	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Demod Test, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	IfLoop0	IfLoop1	x	x	Spare	Spare	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod Test, Write Bytes

Bytes 6-9	x
Bytes 10-11	x
Bytes 12-13	x
Bytes 14-15	x
Bytes 16-17	x
Bytes 18-19	x
Bytes 20-21	x
Bytes 22-23	x
Bytes 24-33	Spare

[IfLoop1-IfLoop0] = Demod IF Loopback, 2b, 0=Disabled, 1=Enabled

Demod Mux, Command [85h]

Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mode	Esc Ovrhead	Mcc Ovrhead	Ratio	EscPort	EscRate	Esc Format	EscDtr
Byte 1	EscDsr	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod Mux, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mode0	Mode1	Mode2	Mode3	EscOh0	EscOh1	EscOh2	EscOh3
Byte 5	MccOh0	MccOh1	MccOh2	MccOh3	Port0	Port1	Port2	Port3
Byte 6	Rate0	Rate1	Rate2	Rate3	Frmt0	Frmt1	DtrMd	DsrMd
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 8	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 9	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 10	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 11	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod Mux, Read Bytes

Bytes 12-13	Mux Ratio X, Unsigned 16b, (1 to 255), X:Y
Bytes 14-15	Mux Ratio Y, Unsigned 16b, (2 to 256), X:Y
Bytes 16-23	Spare

Demod Mux, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mux Mode	Esc Ovrhead	Mcc Ovrhead	0	EscPort	EscRate	Esc Format	EscDtr
Byte 1	EscDsr	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Demod Mux, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mode0	Mode1	Mode2	Mode3	EscOh0	EscOh1	EscOh2	EscOh3
Byte 5	MccOh0	MccOh1	MccOh2	MccOh3	Port0	Port1	Port2	Port3
Byte 6	Rate0	Rate1	Rate2	Rate3	Frmt0	Frmt1	DtrMd	DsrMd
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 8	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 9	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 10	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 11	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod Mux, Write Bytes

Bytes 12-13	X
Bytes 14-15	X
Bytes 16-23	Spare

[Mode3-Mode0] = Mode, 4b, 0=Disabled, 1=Standard IBS, 2=Enhanced IBS, 3=Custom IBS

[EscOh3-EscOh0] = ESC Overhead (Custom Mode Only), 4b, 0=Disabled, 1=300, 2=600, 3=1200, 4=2400, 5=4800, 6=9600, 7=19200, 8=38400
[MccOh3-MccOh0] = MCC Overhead (Custom Mode Only), 4b, 0=Disabled, 1=300, 2=600, 3=1200, 4=2400, 5=4800, 6=9600, 7=19200, 8=38400
[Port3-Port0] = ESC Port, 4b, 0=RS-232, 1=RS-485 2-Wire, 2=RS-485 4-Wire, 3=RS-485 4-Wire Driver On
[Rate3-Rate0] = ESC Rate, 4b, 0=300, 1=600, 2=1200, 3=2400, 4=4800, 5=9600, 6=19200, 7=38400
[Frmt1-Frmt0] = ESC Format, 2b, 0=N71, 1=P71, 2=N81, 3=P81
[DtrMd] = ESC DTR Mode, 1b, 0=Normal (Rcv Flow Control), 1=Ignore
[DsrMd] = ESC DSR Mode, 1b, 0=Normal, 1=Force Active

Demod LNB, Command [86h]
Read Change Flags (Hybrid & L-Band Only)

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	LnbPwr	Reserved	Reserved	IOut	IMax	IMin	Ref	LoFrq
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod LNB, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	PwrEn0	PwrEn1	PwrEn2	RefEn0	RefEn1	RefEn2	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod LNB, Read Bytes

Bytes 8-9	Reserved
Bytes 10-11	Reserved
Bytes 12-13	LNB Current Out, Signed 16b, (0 to >500), 1mA Increments
Bytes 14-15	LNB Current Max, Signed 16b, (5 to 500), 1mA Increments
Bytes 16-17	LNB Current Min, Signed 16b, (5 to 500), 1mA Increments
Bytes 18-23	LNB LO Frequency, Unsigned 48b, (0 to 50,000,000,000), 1Hz Increments
Bytes 24-33	Spare

Demod LNB, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	BucPwr	0	0	0	IMax	IMin	Ref	LoFrq
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Demod LNB, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	PwrEn0	PwrEn1	PwrEn2	RefEn0	RefEn1	RefEn2	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Demod LNB, Write Bytes

Bytes 8-9	X
Bytes 10-11	X
Bytes 12-13	X
Bytes 14-15	LNB Current Max, Signed 16b, (5 to 500), 1mA Increments
Bytes 16-17	LNB Current Min, Signed 16b, (5 to 500), 1mA Increments
Bytes 18-23	LNB LO Frequency, Unsigned 48b, (0 to 50,000,000,000), 1Hz Increments. **Note 1**
Bytes 24-33	Spare

[PwrEn2-PwrEn0] = LNB Power, 3b, 0=Disabled, 1=Enabled, 18 Volts, 2=Enabled, 13 Volts.
[RefEn2-RefEn0] = LNB 10MHz Ref, 3b, 0=Disabled, 1=Enabled
Note 1: When the LNB LO frequency is not zero the Demod IF frequency is converted to an RF frequency.

Interface Status, Command [C0h]
Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	I/O	RTS	CTS	DCD	DTR	DSR	Test	TstBER
Byte 1	SynLoss	Errors	Bits	EFS	ErrSec	TotSec	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Interface Status, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	OnLine	RtsN	CtsN	DcdN	DtrN	DsrN	TerLoop	SatLoop
Byte 5	BerI/O	MPtrnEn	DPtrnEn	DPtnLck	Int'f* Reset	Int'f* Alarm	Int'f* Failure	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Interface Status, Read Bytes

Bytes 8-9	Test BER, 16b, Mantissa (1000=1.000)
Bytes 10-11	Test BER, 16b, Negative Exponent
Bytes 12-15	Test BER Sync Loss Count, 32b
Bytes 16-23	Test Error Count, 64b
Bytes 24-31	Test Bit Count, 64b
Bytes 32-33	Test Error Free Seconds, 16b, (0 to 10000), 0.01% Increments
Bytes 34-37	Test Erred Seconds, 32b
Bytes 38-41	Test Total Elapsed Seconds, 32b
Bytes 42-53	Spare

Interface Status, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	0	0	0	0	0	0	0	TstBER
Byte 1	SynLoss	Errors	Bits	EFS	ErrSec	TotSec	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Interface Status, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	x	x	x	x	x	x	x	x
Byte 5	x	x	x	x	x	x	x	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Interface Status, Write Bytes

Bytes 8-9	x, Write Restarts BER Test
Bytes 10-11	x, Write Restarts BER Test
Bytes 12-15	x, Write Restarts BER Test
Bytes 16-23	x, Write Restarts BER Test
Bytes 24-31	x, Write Restarts BER Test
Bytes 32-33	x, Write Restarts BER Test
Bytes 34-37	x, Write Restarts BER Test
Bytes 38-41	x, Write Restarts BER Test
Bytes 42-53	Spare

Int'f* is Optional Interface status. Optional Interfaces are currently SDMS (Ethernet Type 1), SnIP (Ethernet Type 3) T1, E1 and Drop & Insert.

Interface I/O, Command [C1h]
Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mode	Frmt	RTS	CTS	DCD	DTR	DSR	XData
Byte 1	RData	RClock	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Interface I/O, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mode0	Mode1	Mode2	Mode3	Mode4	Frmt0	Frmt1	Frmt2
Byte 5	RtsMode	CtsMode	DcdMode	DtrMode	DsrMode	XData	RData	RClock
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 8	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 9	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 10	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 11	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 12	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 13	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Interface I/O, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Mode	Frmt	RTS	CTS	DCD	DTR	DSR	XData
Byte 1	RData	RClock	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Interface I/O, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Mode0	Mode1	Mode2	Mode3	Mode4	Frmt0	Frmt1	Frmt2
Byte 5	RtsMode	CtsMode	DcdMode	DtrMode	DsrMode	XData	RData	RClock
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 8	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 9	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 10	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 11	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 12	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 13	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

[Mode4-Mode0] = Interface Mode, 5b, 0=Disable, 1=RS-232, 2=RS-449, 3=RS-449/Unterm, 4=V.35, 5=V.36, 6=EIA-530, 7=EIA-530A, 8 = SDMS*, 9=T1*, 10=E1* *If Installed
[Frmt2-Frmt0] = Async Format, 3b, 0=Disabled, 1=N71, 2=P71, 3=N81, 4=P81
[RtsMode] = RTS Mode, 1b, 0=Normal, 1=Control Mod CXR
[CtsMode] = CTS Mode, 1b, 0=Normal, 1=Force Active
[DcdMode] = DCD Mode, 1b, 0=Normal, 1=Force Active
[DtrMode] = DTR Mode, 1b, 0=Normal, 1=Ignore
[DsrMode] = DSR Mode, 1b, 0=Normal, 1=Force Active
[XData] = Xmt Data Mode, 1b, 0=Normal, 1=Inverted
[RData] = Rcv Data Mode, 1b, 0=Normal, 1=Inverted
[RClock] = Rcv Clock Mode, 1b, 0=Normal, 1=Inverted

Interface Alarm, Command [C2h]
Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	TstAlm	BerAlm	SDMS Alm	Spare	Spare	Spare	Spare	Spare
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Interface Alarm, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	TstAlm0	TstAlm1	BerAlm0	BerAlm1	SDMS Alm0	SDMS Alm1	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 8	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 9	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 10	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 11	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 12	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 13	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Interface Alarm, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	TstAlm	BerAlm	SDMS Alm	0	0	0	0	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Interface Alarm, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	TstAlm0	TstAlm1	BerAlm0	BerAlm1	SDMS Alm0	SDMS Alm1	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 8	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 9	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 10	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 11	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 12	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 13	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

[TstAlm1-TstAlm0] = Interface Test Active Alarm Mode, 2b, 0=None, 1= Alarm A, 2=Alarm B, 3=Alarm A&B
[BerAlm1-BerAlm0] = Interface BER Sync Loss Alarm Mode, 2b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B
[SDMSAlm1-SDMSAlm0] = Interface SDMS Alarm Mode, 2b, 0=None, 1=Alarm A, 2=Alarm B, 3=Alarm A&B

Interface Test, Command [C3h]
Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	TerLoop	SatLoop	BerI/O	ModBer	DemBer	Spare	Spare	Spare
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Interface Test, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	TerLoop	SatLoop	BerIO0	BerIO1	ModBer0	ModBer1	ModBer2	ModBer3
Byte 5	DemBer0	DemBer1	DemBer2	DemBer3	Spare	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 8	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 9	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 10	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 11	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 12	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 13	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Interface Test, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	TerLoop	SatLoop	BerI/O	ModBer	DemBer	0	0	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Interface Test, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	TerLoop	SatLoop	BerIO0	BerIO1	ModBer0	ModBer1	ModBer2	ModBer3
Byte 5	DemBer0	DemBer1	DemBer2	DemBer3	Spare	Spare	Spare	Spare
Byte 6	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 7	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 8	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 9	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 10	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 11	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 12	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 13	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

[TerLoop] = Interface Terrestrial Direction Data Loopback, 1b, 0=Disabled, 1=Enabled
[SatLoop] = Interface Satellite Direction Data Loopback, 1b, 0=Disabled, 1=Enabled
[BerIO1-BerIO0] = Interface BER I/O Direction, 2b, 0=Satellite, 1=Terrestrial
[ModBer3-ModBer0] = Interface Mod BER Test Pattern Generator, 4b, 0=Disabled, 1=2047, 2=2^23-1, 3=Insert 1 Error
[DemBer3-DemBer0] = Interface Demod BER Test Pattern Detector, 4b, 0=Disabled, 1=2047, 2=2^23-1

Interface SDMS, Command [C4h]
Used Only When SDMS Option is Installed
Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	IPAddr	IPMask	MAC	Options	Version	SerialN	Spare	Spare
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Interface SDMS, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Failed	Reset	Alarm	Spare	Spare	Spare	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Interface SDMS, Read Bytes

Bytes 6-9	IP Address, 32b
Bytes 10-13	Network Mask, 32b
Bytes 14-19	MAC Address, 48b
Bytes 20-36	SDMS Options String Terminated with a 00h
Bytes 37-53	SDMS Version String Terminated with a 00h
Bytes 54-57	SDMS Serial Number, 32b
Bytes 58-65	Spare

Interface SDMS, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	IPAddr	IPMask	0	0	0	0	0	0
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Interface SDMS, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	x	x	x	Spare	Spare	Spare	Spare	Spare
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Interface SDMS, Write Bytes

Bytes 6-9	IP Address, 32b
Bytes 10-13	Network Mask, 32b
Bytes 14-19	x
Bytes 20-36	x
Bytes 37-53	x
Bytes 54-57	x
Bytes 58-65	Spare

Interface SnIP, Command [C4h]
Used Only When SnIP Option is Installed
Read Change Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Type	IPAddr	IPMask	MAC	Options	Version	SerialN	L_Mode
Byte 1	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 2	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
Byte 3	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Interface SnIP, Read Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	Selected	Reset	Alarm	Failed	Lmode0	Lmode1	Lmode2	Lmode3
Byte 5	RedEN	OnLine	Spare	Spare	Spare	Spare	Spare	Spare

Interface SnIP, Read Bytes

Bytes 6-23	Ethernet Interface Type String Terminated with a 00h
Bytes 24-27	IP Address, 32b
Bytes 28-31	Network Mask, 32b
Bytes 32-37	MAC Address, 48b
Bytes 38-54	SnIP Options String Terminated with a 00h
Bytes 55-71	SnIP Version String Terminated with a 00h
Bytes 72-75	SnIP Serial Number, 32b
Bytes 76-106	Spare

Interface SnIP, Write Enable Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Type	IPAddr	IPMask	0	0	0	0	L_Mode
Byte 1	0	0	0	0	0	0	0	0
Byte 2	0	0	0	0	0	0	0	0
Byte 3	0	0	0	0	0	0	0	0

Interface SnIP, Write Flags

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 4	x	x	x	x	Lmode0	Lmode1	Lmode2	Lmode3
Byte 5	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare

Interface SnIP, Write Bytes

Bytes 6-23	Not used in write mode.
Bytes 24-27	IP Address, 32b
Bytes 28-31	Network Mask, 32b
Bytes 32-37	x
Bytes 38-54	x
Bytes 55-71	x
Bytes 72-75	x
Bytes 76-106	Spare

[Lmode] = SnIP Interface Link Mode, 4b, 0=Disable, 1=M&C, 2=Bridge, 3=Router, 4=Pass-Thru, 15=Factory Reset.

Table A – FEC Operating Modes

The possible FEC operating modes has exploded in recent years because of the variety of available and common FECs in use. The PSM-500 Series has many current modes and additional ones in development. Below is a table showing the currently available modes depending on modulation.

PSM-500 Series FEC Option Table Rev 1.8 Modem Firmware V 0.77+														
FEC Type	Sel #	FEC Type Option	Sel #	Code Rates Available	Sel #	R-S Option	Modulation Modes (see notes) & Sel #						
							BPSK 0	QPSK 1	OQPSK 2	8PSK 3	8QAM 4	No Use 5	16QAM 6
None	0	N/A	0	N/A	0		●	●	●				
Viterbi	1	Normal	0	1/2	0	●	●	●	●			·	●
				3/4	1	●	●	●	●			·	●
				5/6	2	●	●	●	●			·	●
				7/8	3	●	●	●	●			·	●
		Swap C0/C1	1	1/2	0	●	●	●	●			·	●
				3/4	1	●	●	●	●			·	●
				5/6	2	●	●	●	●			·	●
				7/8	3	●	●	●	●			·	●
		CT	2	3/4	1	●◆	·	·	·			·	●
				7/8	2	●◆	·	·	·			·	●
TCM	2		0	2/3	0	●				●			
TPC	4	Advanced	0	0.453-16k	0		●	●	●	·	●	·	●
				1/2-16k	1		●	●	●	·	●	·	●
				1/2-4k	2		●	●	●	·	●	·	●
				3/4-16k	3		●	●	●	●	●	·	●
				3/4-4k	4		●	●	●	●	●	·	●
				7/8-16k	5		●	●	●	●	●	·	●
				7/8-4k	6		●	●	●	●	●	·	●
				0.922-16k	7		●	●	●	●	●	·	●
				0.950-4k	8		●	●	●	●	●	·	●
		M5 Full	1	1/2	0		●	●					
		*TPC4k Only		3/4	1		●	●					
				7/8	2		●	●					
		M5 Short	2				·	·					
				3/4	1		●	●					
		*TPC4k Only		7/8	2		●	●					
		M5 Legacy	3										
				3/4	1		●	●					
				7/8	2		●	●					
		CT	4	5/16	0		●						
				21/44	1		●	●	●				
				3/4	2		●	●	●	●	●	·	●
		*TPC16k only		7/8	3		●	●	●	●	●	·	●
				0.95	4		●	●	●	●	●	·	●
							·	·	·	·	·	·	·

PSM-500 Series FEC Option Table Rev 1.8 Modem Firmware V 0.77+														
FEC Type	Sel #	FEC Type Option	Sel #	Code Rates Available	Sel #	R-S Option	Modulation Modes (see notes) & Sel #						
							BPSK 0	QPSK 1	OQPSK 2	8PSK 3	8QAM 4	No Use 5	16QAM 6
LDPC	5	Any	0	1/2	0		●	●	●	●	●	·	●
		Block Size		2/3	1		●	●	●	●	●	·	●
		256~16k		3/4	2		●	●	●	●	●	·	●
		Option 0~6		14/17	3		●	●	●	●	●	·	●
				7/8	4		●	●	●	●	●	·	●
				10/11	5		●	●	●	●	●	·	●
				16/17	6		●	●	●	●	●	·	●
							·	·	·	·	·	·	·
S-Tec	6	Any	0	1/2	0		●	●	●	●	·	·	●
		Interleaver		3/5	1		●	●	●	●	·	·	●
		Block Size		3/4	2		●	●	●	●	·	·	●
		Option 0~4		4/5	3		●	●	●	●	·	·	●
				5/6	4		●	●	●	●	·	·	●
				7/8									

Notes:

* TPC4k and TPC16 restrictions apply to that line and Code Rate only.
TPC4k M5 Mode Limits: Rate 1/2 = 2.46 Mbps, Rate 3/4 or 7/8 = 4.92 Mbps.
TPC4k CT Mode Limits: Rate 5/16 = 2.048 Mbps, Rate 21/44 = 3.2 Mbps, Rate 3/4 = 5 Mbps, Rate 0.95 = 6.6 Mbps.
◆ Selecting Viterbi, CT option, Rate ¾ or 7/8 automatically sets R-S into 220, 200, depth 4. R-S can be over-ridden, but is then incompatible.
TPC Advanced modes are Datum Systems proprietary implementations that require the TPC16k option only for the colored lines. They offer superior performance to CT modes.
LDPC2k Mode Limits: Approximately 3.00 Mbps.
LDPC16k Mode Limits: 29.52 Mbps limited by interface and modulation.

Common Notes

All numbers are least significant byte first. All strings terminate with a **[00h]** byte and are 16 characters maximum (not including the string terminator).

The **Mode Byte** must be set to **[05h]** for control of the local modem and **[06h]** to control the remote modem (if local to remote internal control channel available).

If an error occurs, **Bit 7** of the **Status Byte** is set to one. Only one flag in the **Read Change Flags** section is set to one indicating the first write enabled option that generated an error (if any). The **Error/Warning Byte** has the type of error that occurred. If bit 7 of the **Status Byte** is not set the **Error/Warning Byte** is a warning (if any).

Status Byte (Returned in All Responses)

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Byte 0	Offline	AlarmA	AlarmB	InfAlm	UnitAlm	DemAlm	ModAlm	Error

Error Codes

[01h] Request Exceeded Available Limits, Request Aborted.
[02h] Request Exceeded Available Limits, Value Set to Limit.
[03h] Requested Option Not Available, Request Aborted.
[04h] Requested Function Read Only, Request Aborted.
[05h] Requested Frequency Exceeded Total Limits, Request Aborted.
[06h] Bad Request, Not Valid Option Number.
[07h] Option Not Installed, Request Aborted.
[08h] Flash ROM Write Error, Request Failed.
[09h] Write Access Denied Error, Request Aborted.
[0Ah] Requested Option Locked, Request Aborted.
[0Bh] Packet has Incorrect Number of Bytes for Selected Command.
[0Ch] Bad Command, Request Aborted.
[0Dh] Bad Unit Configuration for Selected Option
[0Eh] No Mcc Available, Request Aborted.
[0Fh] Mcc Send Buffer Full, Request Aborted.

Warning Codes

[40h*] Demod Fifo Buffer Exceeded Available Limits, Value Set to Limit.
[80h*] Requested Option Not Active Warning.
[01h] Cxr Enable Request Overridden by Cxr Alarm.
[02h] Bit Rate Changed to Available Limit.
[03h] Demod Not Locked Warning.
[04h] Clock Error, Mod & Demod Bit Rates Not Equal.
[05h] Demod in IF Loopback, Requested Cxr Frequency will be Active After IF Loopback Disabled.
[06h] AUPC Maximum Level Changed to Available Limit.
[07h] AUPC Minimum Level Changed to Available Limit.
[08h] Mod Cxr Level Changed to Available Limit.
[09h] No Remote AUPC Data Available Warning.
[0Ah] Reed-Solomon k Factor Changed to Available Limit.
[0Bh] Reed-Solomon n Factor Changed to Available Limit.
[0Ch] Custom IBS ESC Overhead Changed to Available Limit.
[0Dh] Custom IBS MCC Overhead Changed to Available Limit.

*Can be logically ORed with other warning messages.

END OF PROTOCOL

Notes on Creating a Controller Mechanism

I get occasional questions on writing M&C or controllers to talk to and remotely control either the M5 class or M500 class modems. Here are a few suggestions on making a working controller.

1. The modem returns a response very quickly, usually only a few milli-seconds after the end of your response message. You have to be ready to accept the response immediately.
2. If you are using a PC running Windows, the COM port must be set to a binary mode and not do any manipulation of the incoming data. There are no carriage returns or line feeds that would be used to read a line at a time or an entire response packet.
   Linux computers seem to be even more difficult to get into a pure binary character receive mode. Here is a short snippet showing one way to set the COM1 equivalent in Linux which is "ttyS0". The variable "devicename" in this case is a string set to "/dev/ttyS0". This snippet does not show any actual operations other than setting up the port.

```
fd = open(devicename, O_RDWR | O_NOCTTY | O_NONBLOCK);
  if (fd < 0) {
    perror("Device could not be opened");
    exit(-1);
  }
tcgetattr(fd,&oldtio); // save current port settings
// set new port settings for canonical input processing
newtio.c_cflag = B9600 | CS8 | /*STOPBITS | */CLOCAL | CREAD;
newtio.c_iflag = IGNPAR;
newtio.c_oflag = 0;
newtio.c_lflag = 0;       //ICANON;
newtio.c_cc[VMIN]=1;
newtio.c_cc[VTIME]=0;
tcflush(fd, TCIFLUSH);
tcsetattr(fd,TCSANOW,&newtio);
close(fd);
```

3. The modem's processor is an Infineon C166 derivative, and it writes words in classic Intel or any X86 type "Little-Endian" style. If you are using a controller that is "Big-Endian" like the Motorola/Freescale processors then the byte order of a variable is reversed and will have to be re-ordered before evaluating or sending back to the modem.
4. Calculating checksums is not difficult as long as a computer is doing it. Here is a simple "C" routine to calculate checksums. It could be made considerably simpler, but it has the ability to start and stop anywhere with a string of characters representing a packet.

```
/* ----------------------------------------------
Calculates the 8 bit checksum for modem packets
---------------------------------------------- */
static char checksum8 (char *s, char start, unsigned char len)
{
    unsigned char i, sum8 = 0;
    for (i = 0; i < start + len; i++) {
        if (i >= start)
            sum8 = (sum8 + s[i]) % 256;
    }
    return (256 - sum8);
}
```

5. If you want to define Records or Structures to hold the variables of a particular message, those variables or the whole structure will normally have to be specified as "Packed". This is because most languages will use, for example, a 4 byte integer as their least standard variable size. Packing should allow you to copy the message data directly into the instance of the structure.

```
typedef struct {
  unsigned int  /* Bits little endian aligned */
    Freq:1, Offset:1, Level:1, Output:1,
    Mod:1, Spectrum:1, Mode:1, Preamble:1,

    AUPC:1, AEbNo:1, AMaxLvl:1, AMinLvl:1,
    Mute:1, Imped:1, :2,

    :16;

  unsigned int  /* Bits little endian aligned */
    CxrEn:1, CxrHrd:1, Mod0:1, Mod1:1,
    Mod2:1, SpcInv:1, BurstInst:1, Mode0:1,

    PreLng0:1, PreLng1:1, PreLng2:1, PreLng3:1,
    AUPCEn:1, Mute0:1, Mute1:1, ImpSel:1,

    :16;

  unsigned int   Cxr_Freq;   // CXR Frequency, 32b, (50,000,000
to 90,000,000), 1Hz Increments
  short int Cxr_Freq_Res;  // CXR Frequency, Reserved for L-
BandFrequency Extension
  int   Cxr_Offset;    // CXR Offset, Signed 32b, (-1,250,000 to
+1,250,000, 1Hz Increments
  short int Cxr_Lvl;       // CXR Level, Signed 16b, (-350 to
+50), 0.1dB Increments (dBm)
  unsigned short int AUPC_EbNo;  // AUPC EB/No, 16b, (30 to 200),
0.1dB Increments
  short AUPC_Max_Lvl;  // AUPC Max Level, Signed 16b, (-350 to
+50), 0.1dB Increments
  short AUPC_Min_Lvl;  // AUPC Min Level, Signed 16b, (-350 to
+50), 0.1dB Increments
  short Up_Conv_LO[3]; // Up Converter LO, Reserved for L-Band
  int Spare;

} __attribute__((__packed__)) M5Mod_IF_t;
```